Freespire » Xara » Firefox 2.0 » Gentoo

LXF DVD внутри!



LINUX FORMAT

Главное в мире Linux

Декабрь 2006 № 12 (86)

РАБОТАЕМ С WINE

Запустите любимые Windows-приложения в Linux! с. 46

С днем рождения, Linux!



HotPicks

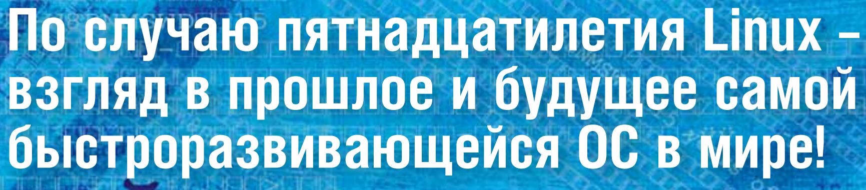
Лучшие новинки открытого ПО на планете с. 112

Карма Kamaelia

Разработка распределенных сетевых систем на Python – проще, чем вы думаете! с. 62

Inkscape vs Xara

Сравнение ведущих инструментов с. 50

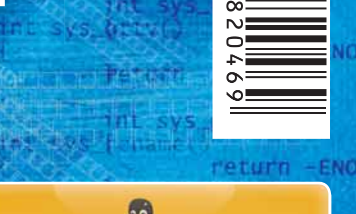


По случаю пятнадцатилетия Linux – взгляд в прошлое и будущее самой быстроразвивающейся ОС в мире!



«В нем есть концепции, которых Ларри, на данный момент, побаивается»

Нат Торкингтон – о Perl и без прикрас с. 54







К Вашим услугам...

Вдохновившись темой номера, мы поинтересовались у Команды LXF: «Как лучше всего отпраздновать 15-летие Linux»?



Пол Хадсон
Заведите пингвина! Не этого открыточно-аппетитного симпатягу, а настоящего живого пингвина в ванной!



Грэм Моррисон
Поеду на день в Бристольский зоопарк ублажать пингвинов кучей селедки!



Майк Сондерс
С бокалом вина, у зажженного камина и за свежеинсталлированной MikeOS 0.22. Кайф...



Эфрейн Эрнандес-Мендоса
Надо напоить Тукса до потери пульса, как сделал бы любой настоящий бойскаут!



Ребекка Смолли
Круглогодичной вечеринкой в Финляндии с фестивалями инсталляций в саунах и GNU Hurd в каждом подарочном носке.



Эндрью Грегори
В старом добром стиле – с ящиком алкогольных коктейлей на задворках кабака.



Нейл Ботвик
Как еще, если не с бесплатным пивом! Свобода слова – тоже хорошо, но обилие пива обычно сильно ее ограничивает...



Д-р Крис Браун
Устроим оргию! Те, кто постарше, могут, впрочем, предпочесть что-нибудь потише...



Дэвид Картрайт
В микропивоварне, за пивом, состав которого можно тут же видеть и... редактировать, если что не так.



Энди Ченнел
Стащите пингвина из зоопарка и пригласите его на пинту-другую пивка. Под рыбку...



Ричард Коббет
С Ричардом Столлменом, выпрыгивающим из торта и убегающим, бормоча: «Нет, ребята, все не так...»



Алекс Кокс
Запустить огромного упирающегося пингвина на животе с гигантской снежной горки. Просто для прикола...

Миссия журнала



- Пропаганда свободного ПО в России
- Продвижение решений с открытым кодом в бизнес-сообществе
- Поддержка российского Open Source сообщества
- Организация трибуны для разработчиков свободного ПО
- Обратная связь между разработчиками и потребителями ПО



15 лет – это срок!

» Год элегантного пингвина подходит к концу. 15 лет, которые потрясли, хорошенько потрясли мир. Следующие 15 могут его перевернуть.

Вспоминается старый анекдот:

Муж и жена засыпают накануне 15-й годовщины свадьбы. Она думает: «Интересно, что он мне подарит завтра?» Он: «Если бы я ее убил в первый день, завтра бы уже освободился»

15 лет – это срок... Мы не знаем, каким был бы мир, если бы юный Линус Торвальдс убил бы (мало ли для этого могло быть разных поводов!) свое детище в младенчестве. Мы не догадываемся, что нам подарит завтрашний день (а судя по новостям этого номера, ожидать можно любого «подарочка»!)

Но мы знаем, что сегодня Linux – то, что объединяет всех нас. Феномен, который обеспечивает нам счастье выбирать, удовольствие ковыряться в коде, общение с такими же чокнутыми, работу (как же без меркантильного интереса!), самореализацию, наконец.

Завтра будет новый день. Новый Год. Новая эра, говорят, тоже будет. Мы ее и начнем. И знаете, хочется увидеть ее такой, чтобы не было мучительно больно за следующие 15 лет, которые мы с вами отсидим. Обязательно отсидим за нашими компьютерами, мониторами, клавиатурами...

За Linux!

Это был тост, господа! :-)

Валентин Синицын и все-все-все мы » Российская редакция LXF
info@linuxformat.ru

Как с нами связаться

Письма для публикации: letters@linuxformat.ru
Подписка и предыдущие номера: subscribe@linuxformat.ru
Техническая поддержка: answers@linuxformat.ru
Проблемы с дисками: disks@linuxformat.ru
Общие вопросы: info@linuxformat.ru
Web-сайт: www.linuxformat.ru
› **Адрес редакции:** Россия, Санкт-Петербург, ул. Гончарная, 23, офис 54.
› **Телефон редакции:** (812) 717-00-37. Дополнительная информация на стр.118



Декабрь 2006 № 12 (86)

Содержание

Весь номер – прямо как на ладони: приятного чтения!



Учебники

Новое лицо Xara.



LXF DVD86

Gentoo Linux 2006.1
Последняя версия Gentoo неплохо подходит для рабочего стола, а LiveCD доставит вам достаточно головоломок, чтобы занять долгие зимние вечера.

PCLinuxOS 0.93
Прекрасное распознавание оборудования, один инсталляционный диск и неплохая подборка ПО: дистрибутив на базе Mandriva ожидает теплый прием!

Linus Torvalds
Нет, это не очередная опечатка. Линус Торвальдс действительно влез на наш DVD, чтобы ответить на главный вопрос всех времен и народов: как произносить слово «Linux»?

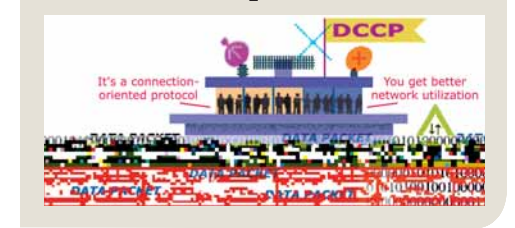
Статьи в формате PDF
Более чем 200 страниц учебников, спецрепортажей и других материалов из прошлых номеров Linux Format!



Gentoo: заточите вашу операционку именно под себя!

Что за штука... DCCP?

Потоковое вещание станет более надежным? Как и почему – на с. 42





LXFHotPicks



PangZero притягателен до ужаса.



ИНТЕРВЬЮ LXF
«Открытый код – это разработчики, которые чешут там, где у них зудит.»
Нат Торкингтон с.34



Hello GTK+ World! с.68
Осваиваем новый инструментарий



Содержание

LXF DVD внутри!

См. страницу 112

Подпишись на Linux Format и сэкономь!



Спецрепортаж

С днем рождения, 

Отметьте пятнадцатилетие открытой ОС в кругу светил Open Source с. 22



А также...

Постоянные рубрики

Week Graphs

Обзоры

› **Если вы никогда не использовали самый старый из существующих дистрибутивов, сейчас самое время попробовать.**

LINUX FORMAT

Новости

ГЛАВНЫЕ НОВОСТИ: Novell + Microsoft = ? » Windows money-back » Mono 1.2 » Java под GPL » Firebird 2.0

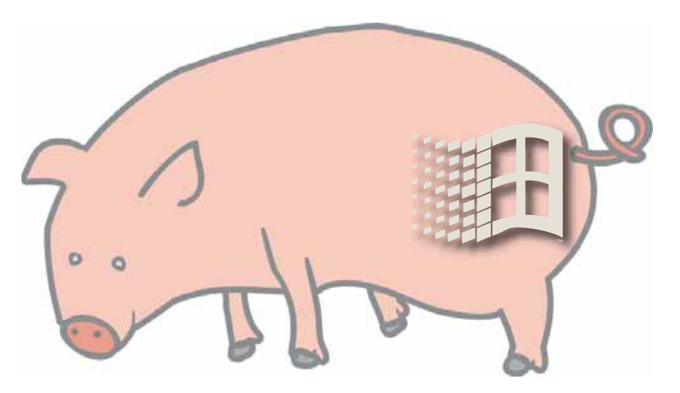
Рубрику ведет
Илья Шпаньков

Зоософтология по-редмондски: купить пингвина, подложить свинью...

Неожиданный поворот в соперничестве Windows и Linux.

Ноябрь выдался поистине сенсационным в плане коренных изменений во взаимоотношениях разработчиков коммерческих Linux-дистрибутивов и их основного конкурента – корпорации Microsoft. Начать с того, что 2 ноября компания Novell подписала с редмондским гигантом соглашение, согласно которому Microsoft выплачивает разработчикам популярной операционной системы SuSE авансовый платёж на сумму 240 млн. долларов в счёт предоплаты за пакет подписных сертификатов на SUSE Linux Enterprise Server. Таким образом гигант IT-индустрии получает право продавать данный программный продукт, а также начинает торговать годовой и многолетней подпиской на обновления и техническую поддержку от компании Novell. Кроме того, в течении ближайших пяти лет Microsoft вложит 60 миллионов в разработку и продвижение на рынок решений виртуализации на базе Linux и Windows, а также потратит 34 миллиона на маркетинговую поддержку всего совместного проекта. Дополнительно соглашение между двумя компаниями подразумевает и взаимную защиту от возможных патентных проблем, в счёт которой Microsoft перечислит новому партнёру 108 миллионов долларов, а Novell в свою очередь выплатит в течении 5 лет не менее 40 миллионов. Участники соглашения также объявили и о начале технического сотрудничества, представленного тремя основными направлениями: виртуализация и полноценное функционирование других операционных систем в «неродной» среде, разработка web-сервисов для управления серверами и повышение совместимости форматов документов MS Office и *OpenOffice.org*.

По мнению некоторых аналитиков, данный неожиданный шаг компания Novell предприняла по причине значительно ухудшившегося финансового положения, но особое недовольство у свободного сообщества вызвал даже не сам факт соглашения с наиболее ярым противником и основным конкурентом свободных операционных систем, а пункт, касающийся патентных урегулирований. Масла в огонь подлил исполнительный директор Microsoft Стив Баллмер [Steve Ballmer], заявивший на конференции Professional Association for SQL Server (PASS) в Сиэтле, что все пользователи и разработчики систем GNU/Linux нарушают патенты Microsoft и должны за это платить, как это сделала компания Novell. Данное высказывание вызвало настоящую бурю протеста со стороны практически всех, кто хоть как-то связан со свободным ПО. Руководитель Novell Рональд Овсепян [Ron Hovsepian] также опубликовал открытое письмо, в котором объяснил данный пункт соглашения лишь желанием привлечь корпоративного пользователя к Linux-системам путём предоставления гарантий в абсолютной легальности свободного ПО. При этом каждая из подписавших соглашение сторон оставляет за собой право предъявлять судебные иски в случае необходимости защитить свою интеллектуальную собственность.



Впрочем, компания Red Hat, позиции которой на рынке уже значительно ослабли из-за недавних действий Oracle и могут ещё более ухудшиться в связи с новым альянсом, выбрала другой способ защиты своих клиентов и объявила, что возьмёт на себя все все заботы в случае, если кто-либо предъявит пользователям программных продуктов Red Hat патентные иски. Кроме того, пытаясь привлечь внимание инвесторов, руководство компании решило разместить свои акции на Нью-Йоркской фондовой бирже, чтобы таким образом улучшить финансовое положение Red Hat. Между тем, в будущем лидера по разработке и продажам коммерческих Linux-дистрибутивов ожидают суровые времена: нетрудно догадаться, что основная цель, которую преследует Microsoft данным соглашением с Novell – желание убрать с рынка наиболее сильного конкурента ожидаемой в 2008 году Windows Server Longhorn. Если редмондскому гиганту это удастся, то всемирно известная корпорация сможет значительно увеличить свою долю на рынке серверных и корпоративных операционных систем, а именно этот сектор приносит основную прибыль. Таким образом, можно констатировать новый виток в борьбе за наиболее выгодную долю рынка ПО и Microsoft сделала в ней свои первые ходы: «приручила» финансами более слабого игрока, поставила в жёсткие конкурентные условия сильного, а остальных «запугала» патентными исками.

http://www.novell.com/news/press/item.jsp?id=1196.



Долгожданный релиз СУБД Firebird 2.0



Объявлен, пожалуй, самый долгожданный за всю шестилетнюю историю, начавшуюся в 2000 году с передачи кода Borland Interbase 6.0 в свободную разработку, релиз открытой СУБД *Firebird 2.0*. Подготовка данной версии продолжалась почти два года и список новшеств и улучшений соответствует затраченному времени. Так, в *Firebird 2.0* была реализована возможность хранения базы данных на raw-устройствах без файловой системы, появилось несколько способов завершения работы с базами данных, увеличен максимальный размер таблицы (до 30 Гб), добавлена поддержка 64-битных платформ (на данный момент только для Linux и для архитектур AMD64 и Intel EM64T), обновлена система резервного копирования данных, а также снят 252-байтовый лимит для поля индексов. Следует отметить, что многие компоненты в данной версии СУБД были практически переписаны «с нуля», что позволило значительно оптимизировать код приложения, а также улучшить производительность и повысить безопасность *Firebird*. Для русскоязычных пользователей особый интерес представляет новый интерфейс для работы с различными кодировками и улучшенная поддержка Unicode.

Несмотря на небольшую задержку в выпуске новой версии, которая ожидалась в середине 2006 года, разработчики обещают не снижать набранных темпов и планируют подготовить следующую версию под номером 3.0, которая будет включать в себя ряд серьёзных изменений, в течении ближайших шести месяцев. Ещё одна причина более быстрой подготовки версии 3.0 кроется в том, что некоторые задумки для третьей версии оказались уже реализованными в *Firebird 2.0*.

http://www.firebirdsql.org/

Новости короткой строкой

» Корпорация Sun Microsystems приняла решение расширить поддержку свободного дистрибутива Ubuntu, предложив его авторам включить в состав системного ПО *Java Enterprise Edition 5*.

» Компания NVIDIA официально объявила о включении в Linux-версию драйверов поддержку AIGLX, позволяющих пользователям открытого ПО использовать возможности видеокарт на уровне, сравнимом с Windows Vista.

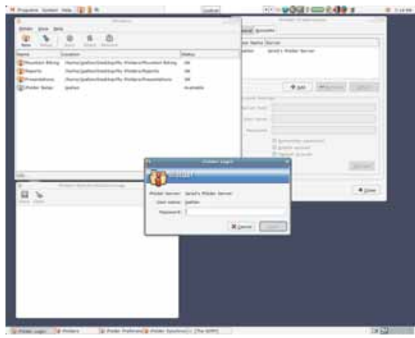
» Ливерморская Национальная Лаборатория планирует значительно улучшить вычислительные возможности своего парка суперкомпьютеров, доведя общую производительность до 100 Тфлопс за счёт приобретения четыре новых Linux-кластеров.

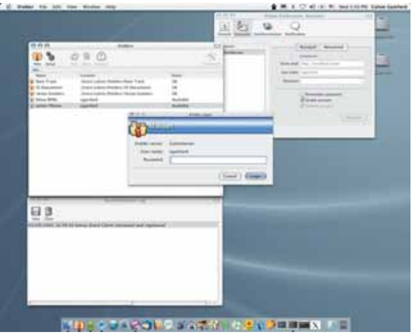
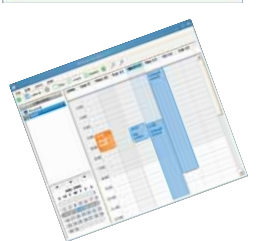
» Компания Linspire открывает для пользователей своих операционных систем несколько бесплатных сервисных служб, включающих почтовый сервис и систему онлайнового хранения данных.

» Open Source Development Labs объявила о включении в дистрибутивы Red Hat Enterprise Linux и SUSE Linux Enterprise от Novell новой версии сетевой файловой системы Network File System v4 (NFSv4).

» Объявлен финальный релиз системы обмена сообщениями D-Bus 1.0.0 под кодовым именем “Blue Bird”, которая призвана стать стандартом для всех Linux-систем.

Mono 1.2 и .NET – курс на сближение



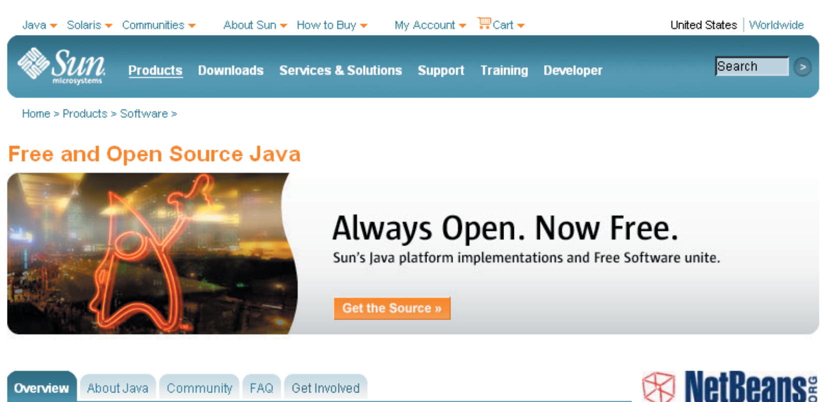
Компания Novell объявила о выходе очередной версии проекта Mono – 1.2. Новый релиз получил несколько значительных изменений, которые способствуют повышению совместимости свободной платформы с проприетарным аналогом .NET. В частности, объявлено о полной поддержке *Windows.Forms 1.1* API, включая полноценную реализацию *System.Drawing*. Поддержка *Windows.Forms* подразумевает полноценную совместимость с WndProc, таким образом большинство виджетов и программ сторонних разработчиков, использующих *WndProc* для добавления специальных эффектов, сможет работать с одинаковой функциональностью в любом системном окружении. В настоящий момент подготовлены драйверы для X11 и Win32, позволяющие полноценно реализовать вышеназванные возможности в любых версиях Windows, а также в системах, использующих в качестве графического сервера подсистему X11 (к ним относятся Linux, Unix, а также OSX при условии использования пакета X11).

Среди других новшеств можно отметить полноценную реализацию System.ServiceProcess, а также базовую поддержку System.Transactions. В новую версию Mono вошли компиляторы C# 1.0 и C# 2.0, которые значительно прибавили в скорости работы по сравнению с прежними реализациями. В добавок ко всему платформа Mono 1.2 стала гораздо неприхотливей в плане процессорных требований: в списке поддерживаемых моделей процессоров теперь представлены 32-битные x86, PowerPC, ARM, s390, SPARC, а также 64-битные x86-64, s390x, Itanium (IA64). Несмотря на тесное сотрудничество с разработчиком .NET компанией Microsoft, проект Mono по-прежнему остаётся свободным программным обеспечением как в плане использования, так и в плане распространения и не будет заимствовать патентованные технологии Microsoft.

http://www.go-mono.com/archive/1.2/

» Начните изучать Mono вместе с Полом Хадсоном в LXF87/88!

Java выходит на свободу



После долгих обещаний компания Sun Microsystems, наконец, приступила к перелицензированию кода Java. В настоящий момент под общественной лицензией GPLv2 опубликованы исходные тексты некоторых компонентов Java Platform Standard Edition (Java SE), включающих технологию Java Hotspot (виртуальная машина Java), компилятор javac и модуль JavaHelp, а также код Java Platform Micro Edition (Java ME), используемый в мобильных устройствах. Также под свободной лицензией теперь распространяются и отдельные компоненты платформы Java Platform Enterprise Edition (Java EE), ранее обладавшие лицензией Common Development and Distribution License (CDDL). В добавок к этому представители Sun заявили, что в 2007 году под свободной лицензией будет выпущен весь код, связанный с Java.

По словам Джонатана Шварца [Jonathan Schwarz], сменившего полтора года назад на посту исполнительного директора Sun Скотта МакНили [Scott McNealy], который слыл ярым противником свободного ПО, новая лицензионная политика компании позволит сделать Java ещё более популярным инструментом разработчиков в области Интернета, а также мобильных, настольных и корпоративных приложений. На сегодняшний день, по данным Sun Microsystems, более 3,8 миллиарда различных устройств в мире в той или иной степени использует Java. Новые инициативы должны ещё более увеличить эту цифру. Достаточно оптимистично встретили изменение лицензионной политики компании и многие известные личности мира свободного ПО. В частности, Тим О'Рейли [Tim O'Reilly] выразил одобрение, смешанное с удивлением: по его словам, все давно ждали от Sun неких шагов в сторону сближения с миром открытого ПО, но никто не предполагал, что шаги эти будут столь решительны.

Между тем, многие аналитики склонны видеть в подобной спешке отнюдь не стремление как можно быстрее влиться в сообщество Free Software по причине осознания его хороших перспектив в будущем, а вынужденную меру, вызванную неравноправной конкурентной борьбой со своим основным оппонентом – корпорацией Microsoft. Также «освобождению» Java в немалой степени могли способствовать и неутешительные финансовые показатели, согласно которым при общей выручке компании в 2006 финансовом году в размере около 13 млрд. долларов, чистый убыток составил 864 миллиона.

Таким образом, выходя на новый уровень взаимоотношений с сообществом Free Software, Sun Microsystems планирует не только привлечь новых пользователей своего программного обеспечения и вернуть мигрировавших на другие платформы, но и открыть дорогу массе сторонних разработчиков, которые смогут внести «свежую кровь» в уже существующее ПО. Этому должны также способствовать и программы Sun Developer Services, которые корпорация Sun Microsystems предлагает программистам, использующим технологию Java и ОС Solaris. Данные проекты предусматривают предоставление разработчикам большого числа справочных руководств, регулярных обновлений ПО, а также осуществление технической поддержки и организацию обучающих курсов, охватывающих весь цикл создания ПО, от разработки до внедрения.

http://www.sun.com/software/opensource/java/

Откажись от Windows – получи деньги

Довольно необычный прецедент (как говорят юристы) был создан в результате действий некоего английского программиста и системного администратора Дейва Митчелла [Dave Mitchell], который, будучи поклонником Linux, решил отказаться от использования Microsoft Windows XP Home SP2, предустановленной на вновь приобретённом ноутбуке от компании Dell. Запротоколировав с помощью фотоаппарата весь процесс загрузки системы вплоть до отказа от принятия условий лицензионного соглашения (EULA), Дейв направил соответствующее письмо в компанию-изготовитель с просьбой вернуть деньги за неиспользуемое ПО. К его удивлению и радости, компания Dell довольно быстро отреагировала на запрос, вернув в общей сложности около 100 долларов. При этом пока данная компания не потребовала вернуть установочные CD и голографическую наклейку с регистрационным кодом Windows.



Надо отметить, что это первый известный случай, когда пользователь получает какое-либо денежное возмещение в результате отказа от использования операционной системы Windows, предустановленной на покупаемом компьютере. Как правило, в ответ на подобные запросы изготовители предлагали возвращать полностью всю покупку, не разделяя её на программную и аппаратную часть. При этом они, по сути, нарушали лицензионное соглашение, прилагаемое к каждой копии Windows. Возможно, пример Дейва станет «пособием» для большого числа пользователей Linux, которые предпочтут сэкономить собственные деньги и отказаться от незаконно навязываемого коммерческого программного обеспечения. LXF





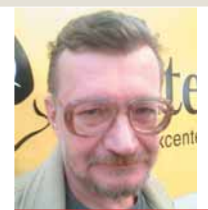


LINUX FORMAT

Обзоры



Новинки программного и аппаратного обеспечения в описании наших экспертов



АЛЕКСЕЙ ФЕДОРЧУК
Свою первую (и последнюю) программу написал еще на Алголе.

Будущее Open Source: коммерциализация или сайентификация?

Этот вопрос широко обсуждается в свете недавних событий, тех самых, что были терты-перетерты как в «бумажной», так и «сетевой» периодике до такой степени, что о них как-то и упоминать уже неприлично. Они, однако, наводят на размышления несколько более общего характера. В частности – а не будет ли вмешательство в развитие Open Source софтверных гигантов началом конца свободного ПО?

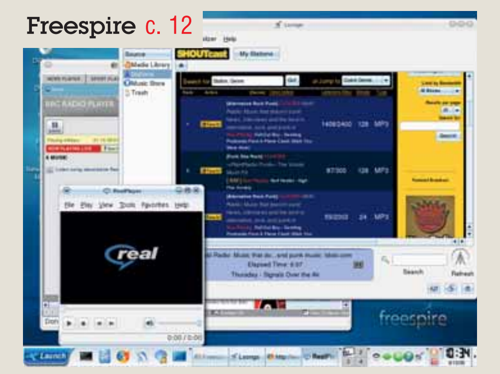
С одной стороны, да – нельзя исключить возможности превращения Linux, точнее, некоторых его дистрибутивов, в сугубо коммерческие, возможно, даже частично закрытые продукты. С другой же – вспомним, откуда начинались и Unix, и Linux, и Open Source вообще? С научных лабораторий, университетов, академических организаций. И люди, его создававшие, никуда не пропадут, да и сферу своей деятельности сменят далеко не все. Так что коммерциализация построенной на Open Source и вокруг него инфраструктуры вполне может вызвать возвращение базовой его части к истокам - так сказать, сайентификацию этого явления.

И тогда Open Source снова, как во времена создания BSD Unix (да и более раннее) будет выполнять свои прямые функции – фундаментальных исследований в области Computer Science, тогда как коммерческие организации займутся прикладными работами и извлечением прибыли из оных. Что ж, так было всегда – одни люди занимались наукой, другие – ее использованием в практических, в том числе и коммерческих, целях...

alv@posix.ru

Сегодня мы рассматриваем...

Freespire с. 12



› Вы, вероятно, слышали, что Click-N-Run стал бесплатным – теперь к нему добавился и бесплатный дистрибутив.

Xara Xtreme с. 10



› Градиенты, текст вдоль кривой и простые, ясные формы – Xara готова к использованию (ну, почти)

НАШ ВЕРДИКТ: пояснение

Все попавшие в обзор продукты оцениваются по одиннадцатибальной шкале (10 – высшая оценка, 0 – низшая). Как правило, мы оцениваем функциональность, производительность, простоту использования и цену, а для бесплатных программ учитывается документация. Кроме того, мы всегда выставляем общую оценку, демонстрирующую наше отношение к продукту.

Выдающиеся решения могут получить престижную награду «Top Stuff». Номинантами становятся лучшие из лучших – просто высокой оценки здесь недостаточно.



Рассматривая свободное ПО, мы обычно указываем предпочтительный дистрибутив. Иногда это означает компиляцию из исходных текстов, но, если разработчики рекомендуют *Autopackage*, мы следуем этому совету.



Google Earth
Разработчик: Google
Сайт: http://earth.google.com
Цена: Бесплатно по закрытой лицензии

Функциональность	10/10
Производительность	9/10
Простота использования	9/10
Документация	9/10

» *Если весь мир – сцена, то Google Earth – театр. Простая в использовании, захватывающая и ободряюще практичная программа.*

Рейтинг 9/10

Firefox 2.0

В Интернет отправилась очередная версия известного браузера. **Алекс Кокс** ознакомился с ней, но острых ощущений не получил.

Вкратце...

» Браузер с открытым исходным кодом. Усовершенствованная поддержка RSS и разумная система вкладок. См. также: *Konqueror*, *Opera*.

По сравнению с прежней версией *Firefox*, 1.5.0.5, круглый номер смотрится гораздо приятнее. Но это отчасти смущает: обычно полные цифры означают радикальные перемены, а здесь ничего такого нет. На первый взгляд, *Firefox 2.0* почти полностью идентичен 1.x. Изменения не видны невооруженным глазом, функции остались прежними, не произошло и включения популярных расширений от пользователей (вроде менеджера загрузок *FlashGot* или скоростного *Fasterfox*), чего вполне можно было ожидать. Движок рендеринга *Gecko* тот же, что и в 1.x, а тест на соответствие стандартам Acid2 выдает изрядную гадость, хотя свежие версии *Konqueror* (OK, мы взяли их в SVN) справляются с ним безупречно.

Конечно, причина этому есть, и вполне весомая – с самого начала была задумана полная совместимость кода *Firefox 2* с 1.5. Все API остались на месте, то есть любое расширение или допмодуль, найденные в Сети, после некоторой подгонки должны заработать с новой версией. Мы опробовали главных подозреваемых (Spotlight, Gmail Space и некоторые темы), и хотя многие отказались работать, конвертированные для новой версии выглядят вполне устойчивыми. Устранять проблемы помогала новая функция Session Restore, безотказно возвращавшая исправную конфигурацию. Крупные перемены в пользовательском интерфейсе и движке рендеринга, похоже, припасены для версии 3 – нынешняя попросту практична.

Firefox никогда не стремился к украшательству: это надежный механизм с незамысловатым интерфейсом. Несмотря на провал Acid2, движок *Gecko* предпочитают многие, и *Firefox* смело может причислить себя к разряду интуитивно понятных программ. Есть в этой версии и ряд не бросающихся в глаза, но приятных добавок, вроде кнопок, вспыхивающих при наведении мыши, или защиты от фишинга (phishing), подсмотренной у *Internet Explorer*.



› Старые расширения работают прекрасно, хотя некоторые надо слегка подогнать.

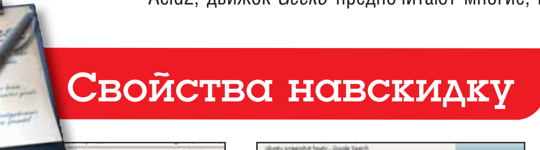
Брать или не брать

Единственное заметное новшество *Firefox 2* состоит в том, что по умолчанию он открывается в окне с двумя вкладками. Такая форма помогает новичкам быстрее освоить работу с ними, а кроме того, намекает на усовершенствованное кэширование: ранее открытые вкладки можно вернуть к жизни двумя щелчками мыши. Приятна манера загружать ссылки в новые вкладки, вместо создания нового окна – экономится и экранное пространство, и кое-какая память.

RSS, возможно, пока интересует немногих, но в новой версии эта функция активно продвигается. Вместе с Live Bookmarks («живые закладки», автоматически обновляющиеся с появлением новых заголовков), вы можете послать заинтересовавшую вас ленту на внешнюю программу просмотра или web-сервис по выбору. Есть дополнительные инструменты для web-разработчиков, включая новый метод интеграции с полем поиска, где *Firefox* уведомляет вас о возможности подключения нового движка для конкретной страницы. Нам понравилась новая система закладок, Live Titles (упрощенный вариант RSS).

Хотя в *Firefox 2* есть немало достойного назваться новшеством, выглядит он слегка невыразительно для столь долго разрабатываемой программы. *Konqueror* и *Opera* набирают скорость, и гонка в самом разгаре; обойдут ли *Firefox* многообещающие соперники с иными методами обзора Web – так же, как мощный *IE* обошел *Netscape* – или отрыв все еще слишком велик? LXF

» Подробнее о разработке *Firefox* см. на стр. 38.



Свойства навскидку

RSS-интеграция

Пользоваться RSS становится все проще – просто щелкните на значке ленты и выберите желаемую цель.



Переход по вкладкам

В меню History (Журнал) появился новый пункт для недавно закрытых вкладок.



LINUX FORMAT Вердикт

Mozilla Firefox 2.0

Разработчик: Mozilla Foundation
Сайт: www.getfirefox.com
Цена: Бесплатно под Mozilla Public License

Функциональность	7/10
Производительность	9/10
Простота использования	9/10
Документация	7/10

» *Один из лучших браузеров на сегодняшний день – но соперники уже наступают на пятки.*

Рейтинг 8/10

Partition Manager 8.0

Поддержка NTFS? Немыслимо! «Всегда-готовый» **Ник Вейч** пробует программу, которая заставит ваш винчестер покрутить цилиндрами.

Вкратце...

» Инструмент для работы с разделами и реанимации рухнувших систем. См. также *SystemRescueCD* или спасательные диски в дистрибутивах.

Винчестеры никуда не годятся. Ну да, не всегда: вообще-то 99.99% времени они являются удобнейшим местом для хранения ваших данных. Вы почти забываете о том, что они есть на свете – пока не случается беда. Тут-то они и подводят.

Эта программа создана именно для подобных случаев. Фактически, *Partition Manager 8.0* не одна программа, а две: приложение для работы с разделами и диск для реанимации. Первая устанавливается только под Windows, поэтому углубляться в нее мы не станем, а вот диск – загрузочный и с Linux, на него стоит взглянуть.

Спасательных дисков для Linux полным-полно: установочный диск практически любого дистрибутива можно употребить для восстановления в случае катастрофы, для этого есть даже несколько специализированных дистрибутивов. Но все Linux-решения наступают на одни и те же грабли: работу с двойной загрузкой.

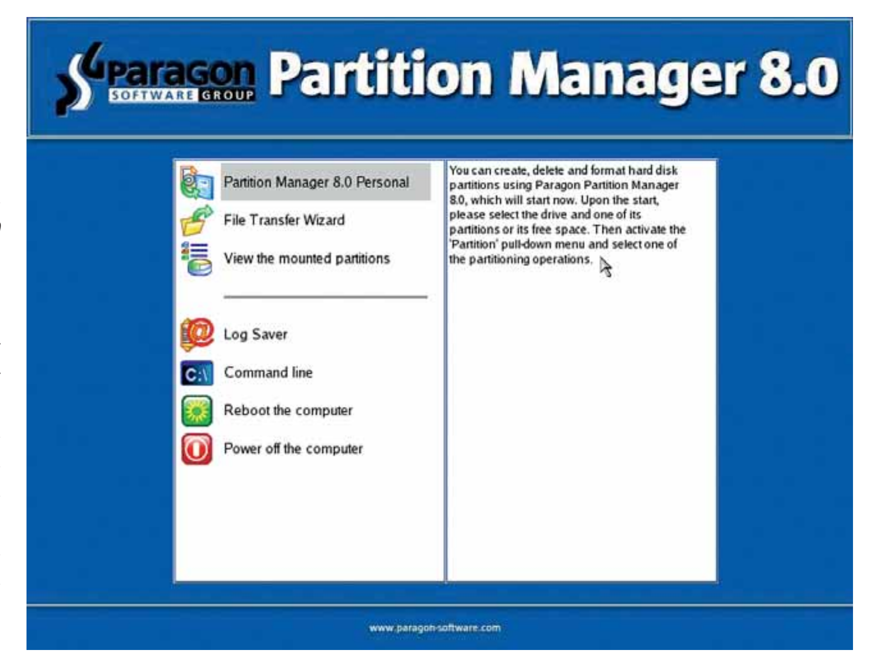
NTFS – во многих отношениях неудобная файловая система, поддержка ее в стандартных ядрах нестабильна, поэтому попытки воскрешения NTFS заранее обречены на провал. Но *Partition Manager 8.0*, с фирменной поддержкой NTFS от Paragon Software, прекрасно перекраивает и клонирует разделы Windows заодно с Linux-овыми. Программа работает также с разделами Reiser и HPFS. Среди полезных инструментов – система копирования файлов, позволяющая восстанавливать их с любого раздела, даже если система не грузится.

Для тех, кто опасается угробить свои данные или не может сообразить, как лучше перераспределить разделы, предусмотрен удобный виртуальный режим: вы заготавливаете список необходимых операций, а потом программа разом их исполняет. Это дает простор для экспериментов, которые трудно провести на «живой» файловой системе – да и боязно!

Если же вы храбрый малый, для вас есть встроенный шестнадцатеричный редактор данных на диске. Он может очень пригодиться для поправки загрузочного кода на Windows-разделах, который нередко отказывается работать, если его потревожат (пробуйте только, если вы уверены в своих силах!).

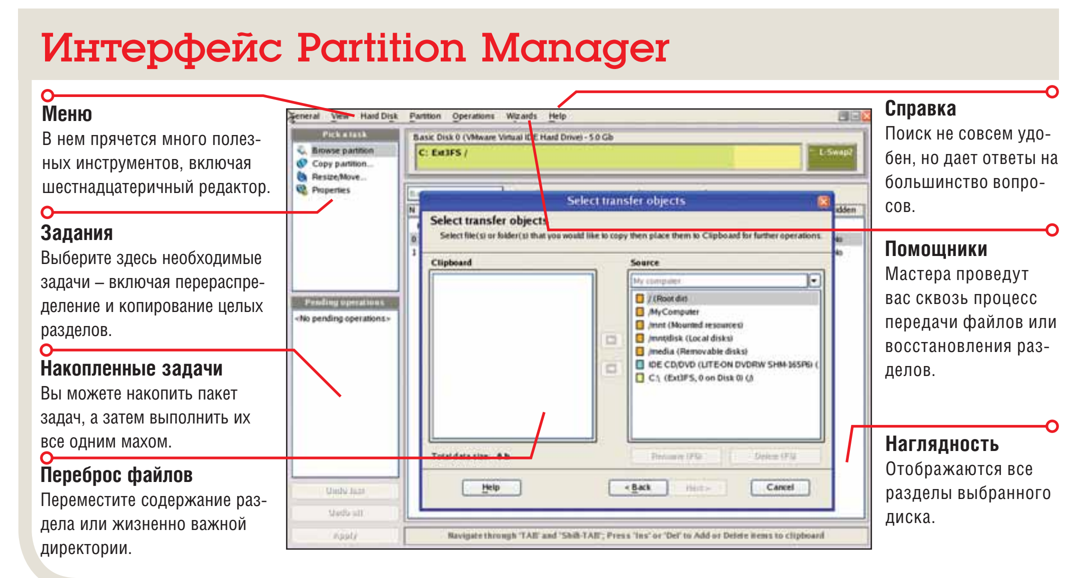
Ограничение *Partition Manager* – он не умеет работать с логическими томами (по крайней мере, на нашем экземпляре Fedora Core не смог). Но, хотя LVM и используется в Fedora по умолчанию, все-таки для большинства настольных машин это экзотика.

В наших тестах программа успешно справилась с перераспределением разделов, копированием их, поиском файлов – словом, со всем, что мы только могли ей предложить – без осечки. Конечно, для чистых Linux-систем средств такого рода хватает, но если на машине стоит еще и Windows, эта программа может стать просто спасительной. LXF



› Спасательный диск быстро грузится и без усилий опознает любой диск или раздел.

Интерфейс Partition Manager



Меню
В нем прячется много полезных инструментов, включая шестнадцатеричный редактор.

Задания
Выберите здесь необходимые задачи – включая перераспределение и копирование целых разделов.

Накопленные задачи
Вы можете накопить пакет задач, а затем выполнить их все одним махом.

Переброс файлов
Переместите содержание раздела или жизненно важной директории.

Справка
Поиск не совсем удобен, но дает ответы на большинство вопросов.

Помощники
Мастера проведут вас сквозь процесс передачи файлов или восстановления разделов.

Наглядность
Отображаются все разделы выбранного диска.

LINUX FORMAT Вердикт

Partition Manager 8.0

Разработчик: Paragon Software Group
Сайт: www.paragon-software.com
Цена: $4,20 (для РФ)

Функциональность	8/10
Производительность	9/10
Простота использования	8/10
Документация	8/10

» *Partition Manager надежен, быстр и прост в использовании – рекомендуем для управления Windows-разделами.*

Рейтинг 8/10

Slackware Linux 11

Спустя почти год со времени выхода последней версии, **Том Уилкинсон** снова посидел на коленях у дистрибутива-дедушки, но не обнаружил там ни ядра 2.6, ни менеджера пакетов.

Вкратце...

» Старожил среди Linux-дистрибутивов, любимый фанатиками и настройщиками config-файлов последовательного порта. См. также: Debian или Gentoo.

Подход Slackware к дистрибуции Linux можно выразить кратко: «Не сломалось – не чини». Старейший из выживших, этот дистрибутив осваивал еще ядро 1.0 (выйдя в 1993 на ядре 0.99), среди его потомков Red Hat, Mandrake и SUSE. Но пока они мчались вперед, Slackware двигался не спеша, только благодаря неоценимым способностям своего единственного попечителя, Патрика Фолкердинга [Patrick Volkerding].

Поэтому инсталлятор за много лет почти не изменился – он все еще текстовый и по-прежнему использует утилиты командной строки в дополнение к пунктам меню. Он примитивен, но все, что нужно, здесь есть, включая подробные инструкции – хотя логика команд не всегда настолько ясна, как хотелось бы. Но это простительно, ведь дистрибутив и не рассчитан на новичков.

Философия становится очевидной сразу после запуска: Slackware – один из немногих главных дистрибутивов, где по умолчанию стоит ядро 2.4. Это выглядит сверхконсерватизмом: ядро 2.6 находится в **/testing** уже как минимум два года, а сейчас в ходе предварительной настройки его можно и установить. По идее, давно пора включить его в основной дистрибутив.



› **Легкий *Xfce* – второе после KDE рабочее окружение Slackware; но, в русле присущей Slackware экономности, можно было бы назначить его и основным.**

> «Несмотря на консерватизм, приложения собраны новейшие.»

Системе пакетов Slackware тоже следовало бы повзрослеть за эти годы. Формат пакетов – проще некуда: это **.tar**-файлы, распаковываемые в корень файловой системы. Нет ни автоматического управления зависимостями, ни (в основном дистрибутиве) системы автоматической загрузки индивидуальных пакетов. Еще пару лет назад это было приемлемо, но сегодня любой известный дистрибутив имеет хоть какую-то систему управления пакетами, а здесь ее остро недостает. Инструмент управления пакетами существует в тестовых сборках уже несколько лет (*slackpkg*), но сверхконсервативная природа дистрибутива и на сей раз не позволила включить его в основной набор.

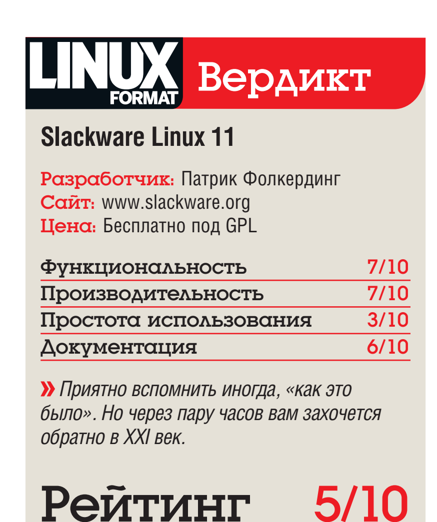
Многие пакеты исключены из версии 11, так как Фолкердинг не в состоянии поддерживать их в одиночку. Крупнейшая из потерь – рабочий стол Gnome, поддержка которого предоставлена сообществу. В тот же список попал и *AbiWord*, и некоторые библиотеки для работы со шрифтами, а некоторые пакеты, и среди них *Apache*, разделены на «библиотеку» и «собственно приложение».

Однако, при всей своей экстравагантной косности, Slackware располагает внушительной коллекцией настольных приложений, среди которых *Amarok 1.4.1*, *KOffice 1.5.2*, *Xine 0.99.4* и изюминки KDE 3.5.4, *Xfce 4.2.3.2* и *Window Maker*. Любопытно, что, наряду с крайним консерватизмом в обновлении ядра, Slackware находится почти на переднем крае по выбору приложений – и включает все больше и больше их с каждой версией. Наш инсталляционный DVD «весит» почти вдвое больше прежних двух CD.

Может быть, эта двойственность и обуславливает слабую востребованность Slackware. Тяга к новейшим версиям одних программ в ущерб другим гарантирует неудовлетворенность множества пользователей. LXF

Архитектурный вопрос

Пакеты Slackware скомпилированы для машин 486 архитектуры, и большинство процессоров x86-класса примут их «на ура». Это означает, что Slackware можно использовать на сравнительно старых машинах в качестве сервера не слишком тяжелого контента, вроде статических web-сайтов. Официально поддерживается только архитектура 486, хотя сообщество предлагает множество иных, включая монстров IBM S\390 и AMD64/EM64T. В настоящее время Фолкердинг не думает, что пользователи улучшат производительность, используя на своих компьютерах пакеты, скомпилированные не для 486, и будет придерживаться этого мнения, пока не убедится в обратном.



LINUX FORMAT Вердикт

Slackware Linux 11

Разработчик: Патрик Фолкердинг
Сайт: www.slackware.org
Цена: Бесплатно под GPL

Функциональность	7/10
Производительность	7/10
Простота использования	3/10
Документация	6/10

» *Приятно вспомнить иногда, «как это было». Но через пару часов вам захочется обратно в XXI век.*

Рейтинг 5/10



Xara Xtreme 0.7

Публикация исходного кода *Xara* в марте 2006 вызвала радостное оживление. Теперь дело движется к 1.0, и **Энди Ченнелл** выясняет, не зря ли все радовались.

Вкратце...

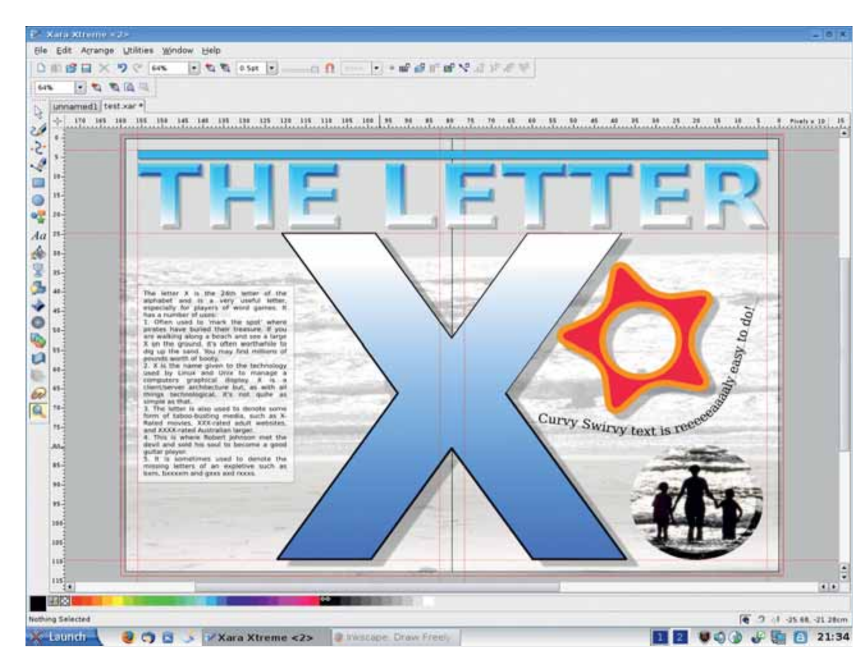
» Редактор векторной графики. См. также *OpenOffice.org Draw* или *Inkscape*.

Прошло свыше полугода с момента публикации исходного кода *Xara Xtreme*, графического пакета для Windows. Тогда многие были настроены скептически, но сейчас, с *Xara Xtreme for Linux 0.7*, разработчики близки к завершению порта и представляют нам первую Linux-версию, которую они сочли «работоспособной». Но она не просто «работоспособна»: *Xara* битком набита изощренными инструментами, которые найдут почитателей в кругах и иллюстраторов, и верстальщиков. Хотя некоторые функции экспорта – в частности, PDF – еще не готовы, приложение в целом выглядит многообещающе.

Для инсталляции *Xara* использует фантастическое средство *Autopackage* (подробнее см. в LXF75), которое славно поработало на нас в Ubuntu, SUSE и Xandros, корректно устанавливая приложения и аккуратно совмещая их с окружением. Интерфейс программы привлекателен и удобен: инструменты общего назначения собраны на левой стороне окна, а контекстно-зависимые функции расположены вверху. Те, кто пользовался *CorelDraw* или *Adobe Illustrator*, почувствуют себя, как дома.

> «Особенно понравилась отзывчивость интерфейса.»

Несмотря на юный возраст, версия вполне стабильна – даже в новейшей сборке. За пару недель ежедневной работы с весьма сложными документами, программа лишь однажды предупредила нас о возможном крахе; затем она сохранила документ и все-таки не рухнула.



› Xara Xtreme справляется и с растровой, и с векторной графикой, быстро обновляя экран.

Особенно понравилась отзывчивость интерфейса. Например, по сравнению с *Inkscape* (на Athlon 2500+ с 512 Мб ОЗУ и видеокартой 128 Мб), приложение запускается и обновляет экран быстрее, а обработка прозрачности, отображение теней и заливка реагируют почти мгновенно. Даже добавка прозрачного градиента поверх шести слоев растрового PNG-изображения, набитого текстом, линиями и прочей графикой, оставила ощущение «реального времени». Панорамирование, масштабирование и редактирование текста, размещенного вдоль кривой, также быстрее.

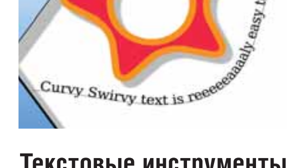
Из минусов: экспортно-импортные операции с SVG пока не работают, печать не развита (хотя графику можно экспортировать и печатать из *Gimp*), а система Live Effects, для подключения расширений *Adobe Photoshop*, нуждается в доработке. Эту и другие функции намечено исправить к версии 1.0, и если разработчики удержат набранный темп – порадуемся их успеху.

Не пропустите

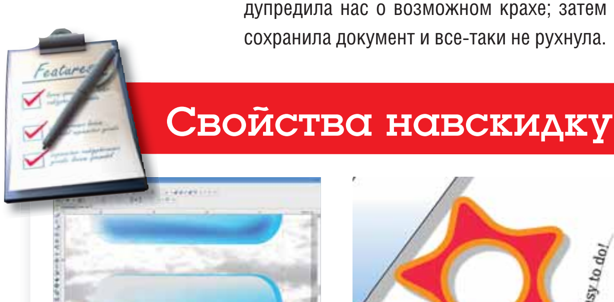
Безусловно, *Xara Xtreme* заслуживает внимания. Завершенные части программы уже сейчас пригодны для создания сложных, изысканных произведений искусства. Работая в связке с *Gimp* и *Scribus*, *Xara* подтверждает способность издательского ПО с открытым кодом выполнять профессиональную дизайнерскую работу, от наброска до конечного продукта. Конечно, *Inkscape* тоже способное приложение – и по оснащенности оно несколько превосходит *Xara*, но обе программы предпочитают работать вместе (ведь исходные тексты все равно свободны), чтобы отвоевать долю рынка у закрытых приложений.

Мы ждем не дождемся выхода версии 1.0, и если разработчики *Xara* сохранят нынешние высокие стандарты, художники и дизайнеры мира Open Source могут расчитывать на счастливый конец в этой истории. LXF

» На стр. 50 приведено сравнение *Xara* и *Inkscape*.

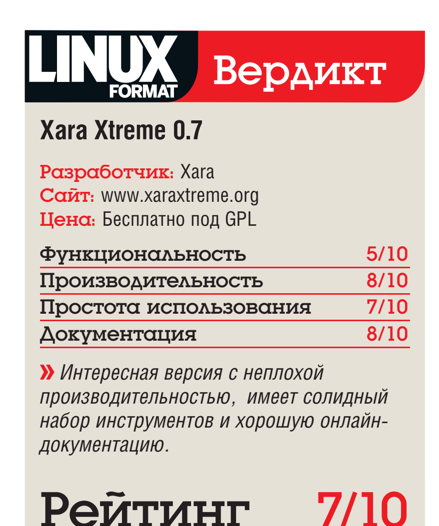


Свойства навскидку



Гладкие текстуры
Можно снабдить фигуры мягкими тенями или подсветкой для создания web-кнопок.

Текстовые инструменты
Инструменты работы с формами весьма эффективны и позволяют, например, пустить текст по кривой.



LINUX FORMAT Вердикт

Xara Xtreme 0.7

Разработчик: Xara
Сайт: www.xaraxtreme.org
Цена: Бесплатно под GPL

Функциональность	5/10
Производительность	8/10
Простота использования	7/10
Документация	8/10

» *Интересная версия с неплохой производительностью, имеет солидный набор инструментов и хорошую онлайн-документацию.*

Рейтинг 7/10

Glade 3.0

Новая команда разработчиков заново написала исходный код. **Грэм Моррисон** окинул дизайнер интерфейсов Gnome свежим взглядом.

Вкратце...

» Графический дизайнер пользовательских интерфейсов, годится для проектирования собственных Gnome-приложений. См. также: *Qt Designer* от Trolltech.

Glade существует уже давно. Как и полагается программе для создания GTK-интерфейса, первая версия *Glade* была доступна почти за год до выхода Gnome 1.0. Но за последние восемь лет *Glade* познал и приливы, и отливы. Команда разработчиков пережила трагическую потерю одного из ведущих программистов, Чема Селорио (Chema Celorio), погибшего во время прыжка с парашютом в 2003 г.

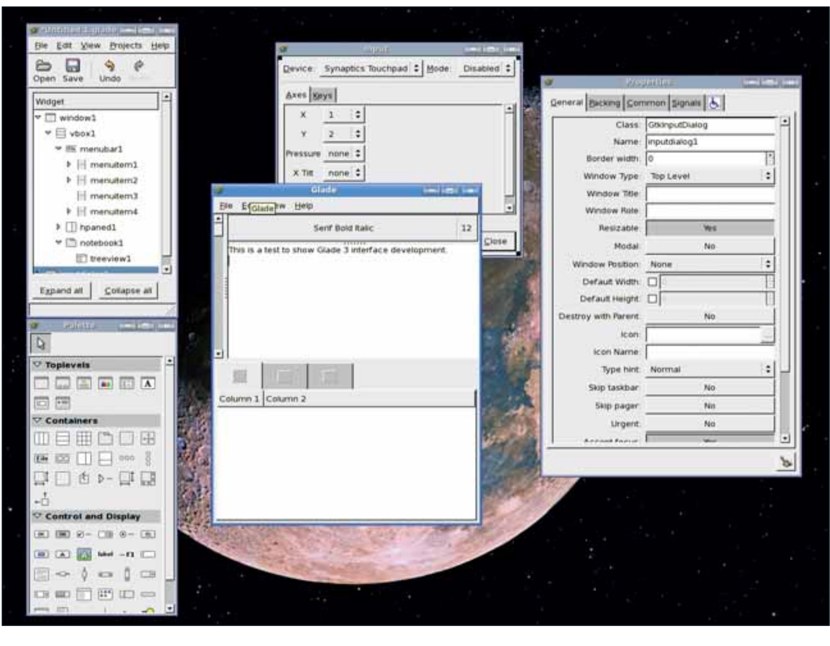
В 2004 г. процесс застопорился, и тогда для продвижения версии 3.0 собралась новая команда. Код *Glade* 3.0 был полностью переписан по сравнению с *Glade* 2.0, планировавшимся к выпуску вместе с *GTK 2*. Перестройка – испытанный способ рационализировать набор функций, совершенствуя проверенное, поэтому новую версию ждали с нетерпением.

Тепло и уют

Пусть 3.0 слегка и засиделась на старте, но вышла как раз вовремя. Gnome наслаждается ренессансом – а что может украсить его лучше, чем инструмент для создания простых и мощных интерфейсов?

Запущенный *Glade 3.0* напоминает скорее *Gimp*, чем инструмент разработчика. Только вот палитра слева предлагает виджеты, а не карандаши, да в плавающих окнах можно изменить свойства виджетов вместо слоев изображения. Но, как и в *Gimp*, разноцветные кнопки искушают вас приглашением творить.

Процесс создания приложений тот же, что и в прежних версиях *Glade*: вы выбираете виджеты в палитре, двигаясь сверху вниз. На самом верхнем уровне выбирается тип окна для приложения. Максимальную гибкость обеспечит простое окно, но можно взять и диалоговое окно, и окно выбора файлов, и окно сообщений. Затем идут контейнеры, управляющие раскладкой виджетов в главном окне. Если впоследствии будет решено разделить окно программы, контейнеры возьмут на себя заботу о различных группах виджетов в отдельных зонах. В последней секции палитры собраны виджеты, отвечающие за функциональность программы. Среди них кнопки, текстовые поля, различные списки, а также индикаторы, календари и выпадающие меню. Работать со всем этим очень просто.

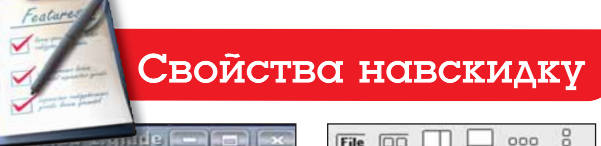


› Пользуясь *Glade*, проектировать интерфейс почти так же просто, как работать в *Gimp*: создаете окно, щелкаете на нужном виджете и задаете нужные свойства.

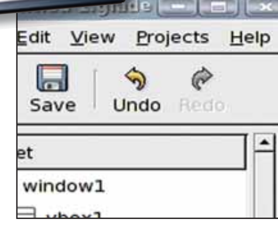
Лес для деревьев

Палитра теперь позволяет переходить между уровнями виджетов более плавно, и можно добавлять собственные каталоги виджетов. Можно путешествовать взад-вперед по стадиям редактирования, пользуясь большими кнопками Undo/Redo в главном окне, причем вы даже не ограничены рамками одного проекта. Можно открывать и редактировать столько окон, сколько душе угодно, и это немаловажно, поскольку приложения редко ограничиваются одним окном. Благодаря объектной системе GObject от *GTK*, отображение виджетов стало эффективнее, и *Glade* работает быстрее своего KDE-аналога, *Qt Designer*. К тому же *Glade* намного проще в использовании: его инструменты для управления компоновкой форм интуитивно понятнее, чем разновесные пружины.

Пока отсутствует интегрированный редактор кода, но есть надежда на скорое включение *Glade* 3.0 в IDE типа Anjuta. *Glade* 3.0 – твердый, уверенный шаг в правильном направлении, укрепляющий позиции Gnome на рабочих столах. LXF

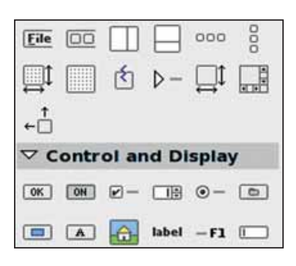


Свойства навскидку



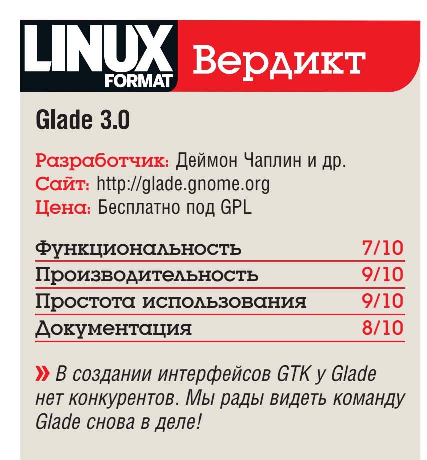
Undo/Redo

С новыми функциями можно экспериментировать без опаски. Есть встроенная контекстная документация.



Усовершенствованная палитра

Новая палитра отлично выглядит и легко приспосабливается к вашему выбору шрифта.



LINUX FORMAT Вердикт

Glade 3.0

Разработчик: Деймон Чаплин и др.
Сайт: http://glade.gnome.org
Цена: Бесплатно под GPL

Функциональность	7/10
Производительность	9/10
Простота использования	9/10
Документация	8/10

» *В создании интерфейсов GTK у Glade нет конкурентов. Мы рады видеть команду Glade снова в деле!*

Рейтинг 8/10

Freespire 1.0

В прошлом выпуске Гаэль Дюваль сказал, что простота использования – еще не гарантия успеха. Но есть ли у Freespire хотя бы это? – спрашивает **Энди Ченнел**.

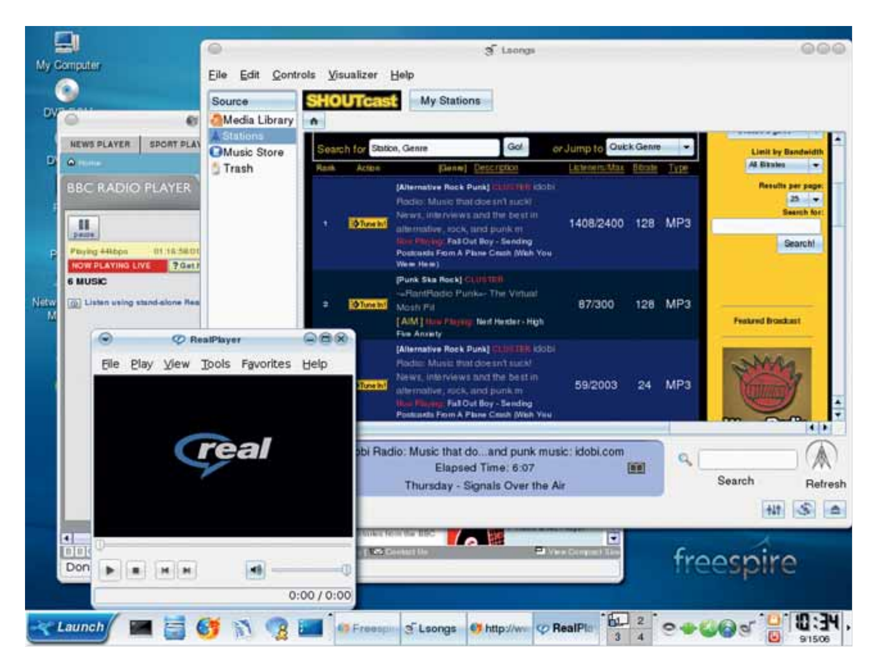
Вкратце...

» Домашний дистрибутив, предназначенный для перехода с Windows. См. также *Ubuntu*, *OpenSUSE* и Fedora Core.

Все знают старую поговорку об осмотре зубов дареного коня. Но что если, порасточав комплименты на упомянутое животное, вы тут же разглядите его увечья? На такие мысли наводит первый выпуск Freespire, нового бесплатного дистрибутива от Linspire. Компания выпустила пару новых дистрибутивов: один – рассматриваемый здесь – содержит подборку закрытых кодеков, приложений и дополнений; другой полностью состоит из свободного ПО. Оба «выводят свободные дистрибутивы Linux на новый уровень простоты», по словам Linspire.

Инсталляция *spire всегда была проста и отражала намерение распознать и настроить все, что можно, с первого же раза и наилучшим образом. Freespire 1.0 – первый из стабильных дистрибутивов, который ведет себя иначе. Наши звуковую и графическую карты ОС определила, но, несмотря на суету разработчиков вокруг закрытых драйверов, приличной работы достичь не удалось. Три года назад разрешение 1024x768 могло бы угодить всем, но наша тестовая машина со 128 МБ видеопамяти готова и жаждет поддержать гораздо более высокое разрешение, а монитор (оставшийся неопознанным) – отобразить его. Обе сетевые карты были настроены неверно (первый случай за все время испытаний дистрибутивов с 2001 г.). А «простое в использовании» хранилище Click-N-Run (CNR) – это просто свалка ядер случайных версий (кому нужна 2.4.16?), старых драйверов и прочего хлама.

> «Мы вдруг заскучали по тонкой изощренности Windows XP.»



› Freespire поддерживает множество медиа-форматов прямо «из коробки».

Поработав несколько часов со Freespire – особенно с темами *Firefox/Konqueror* – мы вдруг заскучали по тонкой изощренности Windows XP. Падающие тени умиляют, но значки угловаты, организованны беспорядочно и действуют с точностью до наоборот к ожидаемому. Правда, темы, расширения и значки можно поменять, но в дистрибутиве, предназначенном для упрощения и облегчения работы, они должны как минимум вести себя предсказуемо.

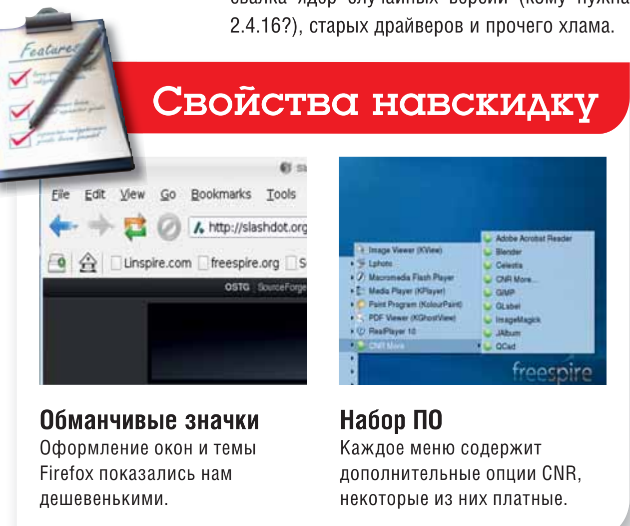
Недогруз

Ограниченность набора приложений в Freespire – это само по себе неплохо: новичок не заблудится в чаще новых функций. А вот организация меню слишком громоздка. Пока вы находитесь в пределах стандартных каналов CNR, все вроде нормально – но стоит выйти за эти пределы, и начинается неразбериха.

По словам разработчиков, этот дистрибутив уникален включением различных закрытых кодеков и приложений, типа *Adobe Acrobat*, Flash, MP3, *QuickTime* и Windows Media. Кодеки-то есть, а вот уникальность спорна: в Xandros то же самое реализовано гораздо проще, а *EasyUbuntu* предлагает больше опций (и надежнее устанавливает драйверы Nvidia).

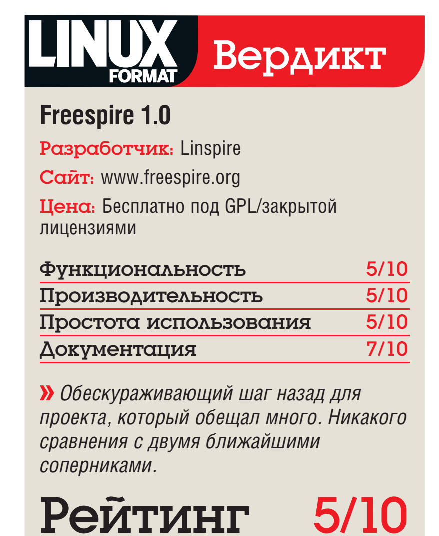
На фоне житейской мудрости Ubuntu, простоты Xandros и даже прежних версий Linspire, Freespire изрядно разочаровывает. Некоторые наши претензии можно было бы отнести на счет оборудования, да только наш тестовый компьютер содержит вполне рядовые комплектующие, и их корректно опознавали и настраивали все дистрибутивы, которые мы пробовали раньше. Интересно сравнить Xandros, Ubuntu и Freespire, ведь все три проекта нацелены на одну и ту же категорию пользователей, осаждая ее с разных сторон. Freespire оказался посередине – пытаясь соединить этическую чистоту Ubuntu и воздушную легкость Xandros, он не достиг ни того, ни другого. LXF

Свойства навскидку



Обманчивые значки
Оформление окон и темы Firefox показались нам дешевенькими.

Набор ПО
Каждое меню содержит дополнительные опции CNR, некоторые из них платные.



LINUX FORMAT Вердикт

Freespire 1.0
Разработчик: Linspire
Сайт: www.freespire.org
Цена: Бесплатно под GPL/закрытой лицензиями

Функциональность	5/10
Производительность	5/10
Простота использования	5/10
Документация	7/10

» *Обескураживающий шаг назад для проекта, который обещал много. Никакого сравнения с двумя ближайшими соперниками.*

Рейтинг 5/10

Сравнение



Каждый месяц мы анализируем для вас тысячи программ – а вы можете отдохнуть!

Файловые менеджеры



Файловый менеджер – всего лишь менеджер файлов, так? Не совсем, говорит **Майк Сондерс** после мега-исследования функций, предлагаемых лучшими файловыми менеджерами Linux...



Paul Blachford

О тесте...

Мы провели тест на ПК с частотой процессора 1.5 ГГц и 256 МБ ОЗУ. Критерии:

» **Производительность** Сколько времени требуется для загрузки или просмотра «больших» каталогов?

» **Использование памяти** Чтобы поставить всех в равные условия, все программы были протестированы на одном и том же каталоге.

» **Стабильность** Замечены ли в программе какие-нибудь глюки или отказы в работе?

» **Простота использования** Этот пункт требует более глубокого рассмотрения. Мы решили приблизиться к реальности и попросили нашего художественного редактора проделать различные операции в каждом файловом менеджере, например, переименовать файл. Эффи – идеальный пример среднего пользователя: он работает с компьютером каждый день, но не знаком ни с одним из файловых менеджеров Linux. Поэтому мы замерили, сколько времени у него займет совершение обычных операций над файлами. Таким образом можно судить, насколько проста в освоении каждая программа.

Наш выбор

Согласимся сразу: файловые менеджеры – это не круто. С ними не блеснешь, они не привлекательны и не имеют супер-функций, которыми можно хвастать. Зато они играют огромную роль в жизни 99% пользователей компьютеров (кроме тех, кто все делает из командной строки *Bash* или вручную редактирует шестнадцатиричные образы дисков – ну, таким уж медаль хакера на грудь). От каждодневного труда файловых менеджеров зависим мы все, хоть и не задумываемся об этом: они – своеобразные сантехники системы, наводящие порядок в наших растущих объемах информации.

Файловые менеджеры все больше привязываются к рабочим средам: у KDE есть могучий *Konqueror*, а у Gnome – уважаемый *Nautilus*. Но независимо от вашей рабочей среды или оконного менеджера, выбор файловых менеджеров всегда широк: он может быть быстрым и небольшим или многофункциональным и всемогущим. Сообщество Open Source разработало большое число файловых менеджеров; некоторые рассчитаны на широкое использование, а некоторые только тешат самого разработчика.

Для этого **Сравнения** мы выбрали восемь самых известных претендентов, и оценили их по таким ключевым критериям, как производительность, простота в использовании и функциональность. Одни файловые менеджеры служат продвинутым пользователям: в них есть букеты горячих клавиш и быстрый доступ к командным интерпретаторам, а другие (в частности, под KDE и Gnome) нацелены на новичков и привечают виджеты и меню. Наша подборка охватывает интересы различных групп пользователей: включен даже файловый менеджер, работающий исключительно в текстовом режиме.

Итак, без лишних слов, посмотрим, как выстроить наши менеджеры «по росту». Мы собрали массу статистики и фактов о производительности каждой программы в таблицу (вы найдете ее конце статьи), а в каждом мини-обзоре проанализировали программу в целом. Кроме того, все рассматриваемые программы есть на DVD, и вы можете сами их испробовать. Файлы наготове...

Xfm

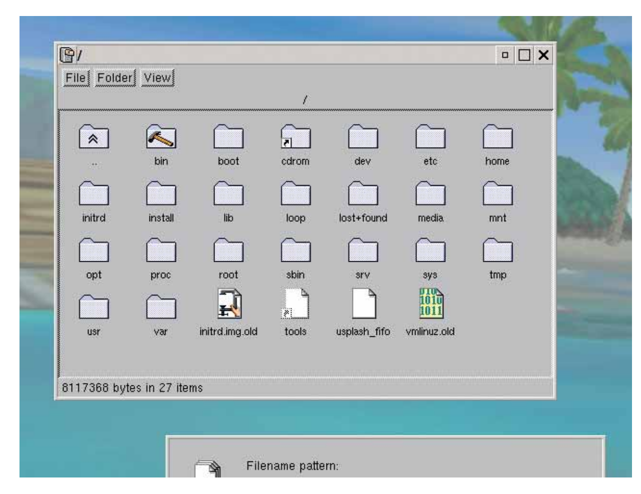
Работает на *X*. Файловый Менеджер. Это *Xfm*!

ОК, имя программы не блещет выдумкой. Но простим это оригинальным разработчикам: *Xfm* появился еще в ранние девяностые, когда не было ни *Konq*, ни *Nauti*, а файловые менеджеры для Unix были в новинку (большинство пользователей бойко выстукивали символы в командной строке). Удивительно, но *Xfm* дожил до наших дней, и включен в репозитории многих дистрибутивов. Ваш автор сам регулярно пользовался им в конце 90-х – по компьютерному возрасту, это древний старец, но он все еще имеет некоторую ценность.

Xfm исключительно нетребователен и использует только инструментарий *X* (*Xaw*, возможно, с простенькими 3D-эффектами, если у вас установлены библиотеки). Благодаря этому налицо преимущество в скорости, использовании памяти и установке, но, откровенно говоря, выглядит *Xfm* по сравнению с современными программами на *GTK* и *Qt* ужасно. Топорные виджеты и серый фон не добавляют шарма программе!

Программа состоит из двух окон, одно из которых служит для навигации по файловой системе, а другое – для запуска программ, здесь заготовлены ссылки на устаревшие приложения (типа *NEdit* и *XCalc*). Другой индикатор возраста *Xfm* – файловые иконки: например, tar-архивы изображены в виде лент, напоминая о днях, когда повсеместно применялись ленточные накопители.

Xfm поддерживает все основные операции с файлами, а также режим drag-and-drop, и имеет возможность просмотра файлов в виде списка. Менеджер довольно хорошо настраивается при условии, что вам не составляет труда отредактировать файлы настройки. И здесь кроется самая большая проблема *Xfm*: при наличии времени, вы можете настроить его так, что он будет вести себя как современный файловый менеджер, но это слишком трудная работа для большинства из нас. Он всегда будет выглядеть слишком архаично и требовать много внимания. Отсутствие дополнительных функций и интеграции с рабочим столом позволяет ему достичь огромной скорости, поэтому он является неплохим выбором для очень слабых машин или небольших дистрибутивов, но для современных настольных компьютеров он слишком стар.



› **Сейчас он смотрится как непропеченный хлеб, но в ранние девяностые для X это было вполне прилично.**



Nautilus

Приложите ухо к этой ракушке – и услышите GMC.

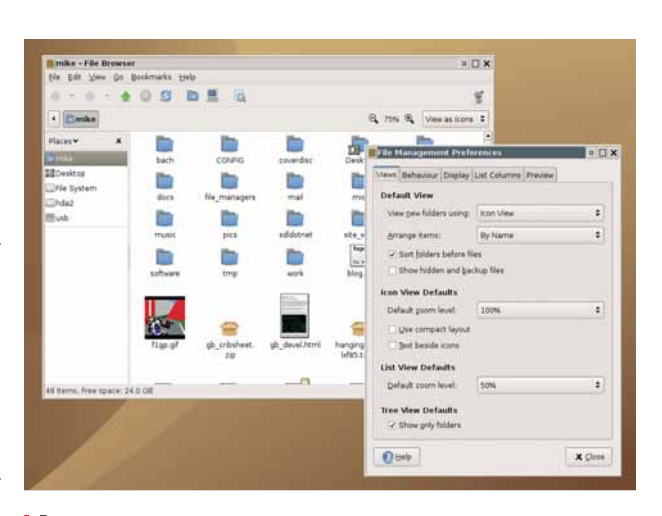
До версии Gnome 2.0 в этой рабочей среде использовался менеджер GMC, *GTK*-вариация программы *Midnight Commander* (*см. стр. 16*). *Nautilus* был разработан ныне покойной Eazel Inc как замена GMC и сразу же подвергся острой критике за медлительность и расход памяти. За последние несколько лет производительность улучшена – но все же это самая прожорливая программа в нашем **Сравнении**.

Nautilus следует философии простоты управления Gnome, имеет опрятный вид и минимальный набор кнопок. Удобный список слева позволяет перемещаться между наиболее используемыми каталогами файловой системы, а также раскрывать и закрывать каталоги. Превосходно организован предпросмотр: изображения на экране показываются в миниатюре, а благодаря *GtkHTML* можно видеть и мини-снимки HTML-страниц, не открывая их. Другая приятная возможность – добавление «эмблем» к файловым иконкам: можно, например, пометить файл бомбой, чтобы его зря не лапали!

В нашем тесте на простоту использования *Nautilus* победил всех: все операции с файлами выполнялись очень легко. Контекстные меню показывают ровно столько каждодневных задач, сколько нужно, чтобы не запугать пользователя их количеством; кроме того, панель навигации наверху показывает ранее посещенные каталоги, и это элегантнее, чем Windows-подобные раскрывающиеся деревья. *Nautilus* поддерживает SMB и SSH, так что можно перетаскивать файлы по сети.

Наше главное нарекание к *Nautilus* состоит в том, что он тормозит и съедает много оперативной памяти – больше, чем мощный *Konqueror* (*см. стр. 17*), что немного настораживает; а возможность предпросмотра файлов может сильно снизить скорость работы при просмотре каталога с большим числом файлов (впрочем, ее можно отключить). На современных компьютерах оперативная память особо не лимитирована, но для старых систем лучше присмотритесь к *Rox-filer*.

> «Можно даже видеть миниатюры HTML-файлов, не открывая их.»



› **Вразумительное окно настроек поделено на вкладки, и не надо сновать по одному огромному экрану.**



Midnight Commander

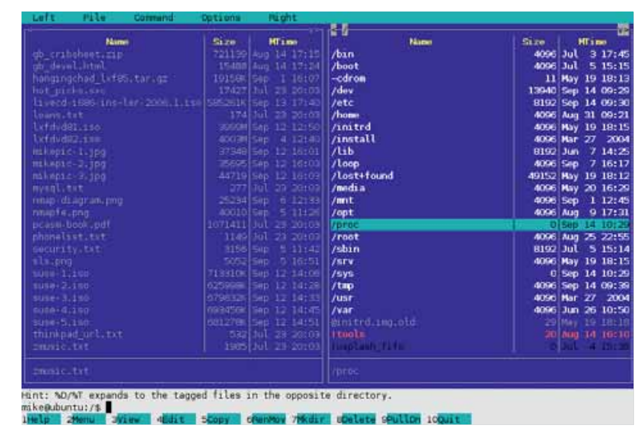
Режим текстовый, мыши нет – бейте по клавишам.

Midnight Commander, всем известный как *MC* – единственный файловый менеджер в нашем **Сравнении**, работающий в текстовом режиме. Многим пользователям управление файлами без использования мыши абсолютно чуждо, но это вполне осуществимо [в принципе, Midnight Commander поддерживает и GPM, – прим. ред.].

MC нацелен на старожилов в Unix, легко запоминающих клавиши быстрого доступа и клавиатурные сокращения. Здесь применен традиционный двухпанельный подход: переключение между панелями происходит с помощью клавиши Tab. В качестве приятного сюрприза, вы можете вызвать клавишей F9 меню, через которое выполняются различные операции над файлами и различные настройки. Один из лучших инструментов *MC* – встроенный командный интерпретатор: в любой момент вы можете ввести обычную текстовую команду (например, переименовать файл) и сразу же увидеть результат. Если вы завсегдатай командной строки, то почувствуете себя комфортно.

Основные операции над файлами (копирование и удаление) доступны через функциональные клавиши, перечисленные внизу экрана, но сжатость названий выбила нашего тестера из колеи: он не догадался, что 'RenMove' означает операцию переименования/перемещения. Зато *MC* включает простенький текстовый редактор, и не надо помнить хитрые комбинации клавиш, чтобы загрузить *Emacs* или *Vim*.

Midnight Commander непрост в освоении для среднего пользователя и не содержит графических изысков, но именно благодаря их отсутствию развивает огромную скорость – он незаменим при подключении через SSH на сервер, когда манипулировать файлами надо очень быстро. *MC* остается лучшим текстовым файловым менеджером, и для серверов и систем, не оснащенных *X*, это самое скоростное и надежное решение.



› **Графический интерфейс пользователя – кому это надо? Идеальный для мышефобов *Midnight Commander* вмещает очень много даже в ANSI-терминалы 80x25.**



Midnight Commander

Версия: 4.6.1

Сайт: www.ibiblio.org/mc

Цена: бесплатно по лицензии GPL

» *Тяжелее в освоении, чем его конкуренты с графическим интерфейсом, но необычайно надежен и быстр.*

Рейтинг 6/10

Gentoo

Дистрибутив Linux для лихачей! Ой, нет...

Нет, с одноименным дистрибутивом *Gentoo* не роднит ничего, разве что жажда скорости. Файловый менеджер *Gentoo* подражает дизайну старого *Norton Commander*, экран которого разделен на две панели, и каждая отображает файловую систему. Идея состоит в том, что вы видите, куда передаете файлы: например, выбрали файлы на панели слева, выбрали каталог справа, а затем жмите на кнопку Скопировать.

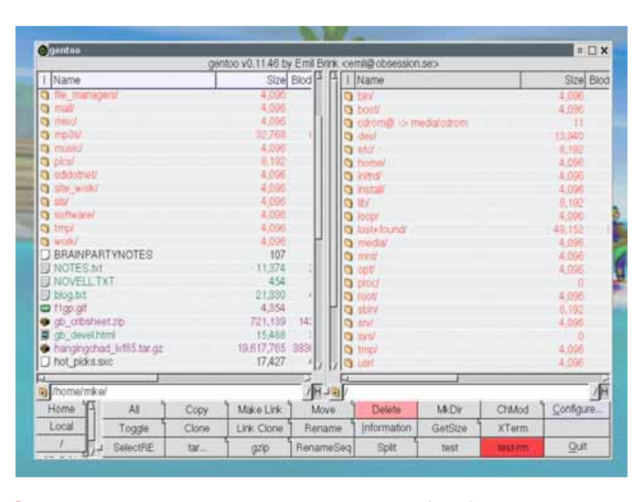
Необходимость ручного выбора отдельных файлов озадачила нашего тестировщика. В программе не предусмотрен режим перетаскивания; вместо этого надо помечать каждый файл. К счастью, внизу окна расположены кнопки, предоставляющие доступ ко всем основным функциям, и у большинства из них понятные имена (хотя имя 'ChMod' может смутить незнакомых с командным интерпретатором).

Gentoo построен на базе *GTK1*, и в нем отсутствует антиалиасинг и другие эффекты, позволяющие вписаться в современные рабочие среды Gnome или Xfce. Однако интерфейс хорошо продуман с точки зрения продвинутых пользователей: имена файлов раскрашены в зависимости от содержимого, предоставляется обильная статистика и множество клавиш для быстрого доступа к функциям.

Несомненно, наиболее впечатляющая черта *Gentoo* – его настраиваемость. Нажмите на кнопку Configure, и увидите целую вселенную опций: от точного расположения окна и способа отображения кнопок до собственных команд и выбора столбцов отображения файлов. Опция «распознавание файлов» распознает столько их типов, что не видя этого самому, даже и поверить трудно! Если вам что-то не понравится в *Gentoo*, вы наверняка сможете это изменить.

Каждому первому этот двухпанельный менеджер с непревзойденным уровнем настройки не порекомендуешь, но если вы уже используете Linux и находите *Konq* или *Nautilus* слишком ограниченными, то это ваш выбор. Эх, была бы версия на *GTK2*...

> «Вы найдете целую вселенную опций.»



› **Удачное использование цветовыделения, но *GTK1*? Фи....**



Gentoo

Версия: 0.11.46

Сайт: www.obsession.se/gentoo

Цена: Бесплатно по лицензии GPL.

» *Необычайно гибок в настройке. Если у вас ностальгия по Norton Commander, это то, что вам нужно.*

Рейтинг 8/10

Konqueror

После Explorer и Navigator, KDE дарит...

K*onqueror* появился в KDE 2.0 как замена *KFM*, предлагая не только файловый менеджер, но и полнофункциональный браузер. Как и многие другие программы KDE, он критикуется за функциональный и визуальный перегруз; мы же не считаем это большой проблемой, потому что любой пользователь (и, понятно, разработчик дистрибутива) легко может как сделать главное окно чудовищно навороченным, так и «опустошить» его до нужного уровня.

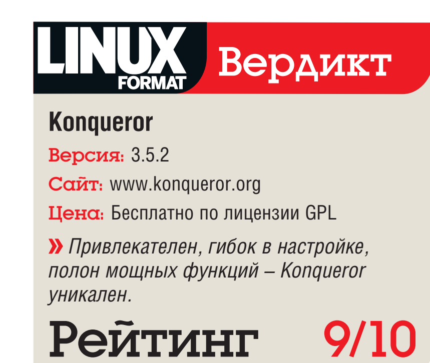
Вне KDE *Konqueror* требует довольно много времени на запуск (около 15 секунд на нашем оборудовании), в первую очередь из-за предварительной загрузки необходимых компонентов KDE. Для типичной сессии ему нужно около 25 МБ оперативной памяти (резидентно); меньше, чем для *Nautilus* (30 МБ), но он заметно медленнее при открытии больших каталогов. Имеется возможность предпросмотра изображений и HTML-файлов, а также превосходное контекстное меню, свое для каждого типа файла.

Konqueror хорошо смотрелся в нашем тесте на дружественность к пользователю: единственной трудностью стало создание нового каталога (наш «проверяющий», понятное дело, искал кнопку, не полагая пункт Create New в контекстном меню подходящим способом). Новичка способно запугать окно Настройки: в нем полно опций, касающихся браузера, и пунктов с названиями типа Получение метаданных по протоколу.

Однако, поддержка просмотра каталогов с использованием огромного количества сетевых протоколов (SSH, SMB, NFS, WebDav и т.д.) дает *Konqueror*'у убойную силу. Благодаря системе KDE KIOSlave отпадает необходимость запуска нескольких инструментов сразу, и для многих пользователей Unix это просто находка. Вы даже можете просматривать на лету содержимое архивов. Будучи правильно настроен, *Konqueror* представляет идеальный баланс возможностей и здравого смысла – что и отличает хорошие программы.

> «Он может быть чудовищно сложным или максимально голым, на ваш вкус.»

› Желая придраться, мы сказали бы, что настройки *Konqi* перемешаны с опциями web-браузера, а это осложняет дело.



Nao

Зато его легко установить.

N*ao* означает «Не Просто Очередной» файловый менеджер; он написан на основе набора компонентов *Fox Toolkit*. Это придает ему определенную энергичность, так как набор инструментов Fox известен хорошей производительностью, но оборотная сторона медали – инородный вид на рабочем столе с приложениями *GTK* и *Qt*. Сюда можно добавить некоторые странности интерфейса: кнопка для поднятия вверх по дереву каталогов представлена в виде стрелки, направленной вниз, а коварная кнопка «Прочее» слева на вид не делает ничего, но срубает все выполненные настройки.

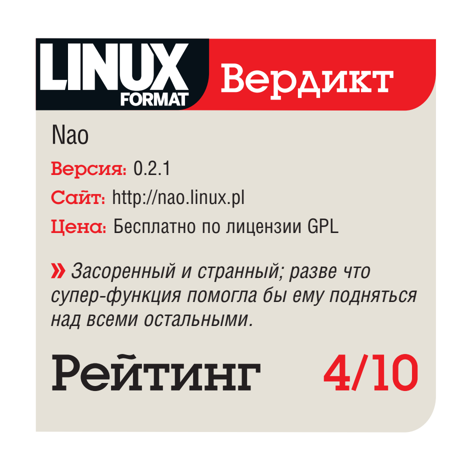
Зато установка простейшая. Не надо мучиться с зависимостями *Fox* и подобными вещами: просто запустите «универсальный двоичный пакет с установщиком», а тот скопирует статически скомпонованный файл в **/usr/bin**. *Nao* обеспечивает некоторый предпросмотр графических файлов, можно также работать в режиме двух панелей (грустно, но режим перетаскивания на данный момент не поддерживается). Большинство операций над файлами доступно из контекстного меню по правому щелчку, но удачи вам, если вы хотите заставить *Nao* делать то, что вам нужно – стиль работы у него омерзительный.

Например, щелкаем правой кнопкой на файле и выбираем Copy [Скопировать]. Что произойдет? Появится миленькое окошко с сообщением «Операция не удалась». Ага, вам следует выделить его в одной панели, в другой – выбрать каталог, затем нажать правую кнопку мыши, выбрать операцию копирования и посмотреть, заработает ли все это. У нашего тестера было мало проблем с простыми операциями (*см. таблицу на стр. 19*), но выполняя что-нибудь посложнее, вы не получите ничего, кроме головной боли.

Цель *Nao* – быть «самым конфигурируемым», и конечно же, в диалоге настройки присутствует большое число опций. Справедливости ради, отметим, что программа достигла лишь номера 0.2.1, и много чего еще предстоит сделать, но до тех пор, пока не будет переделан графический интерфейс главного окна и диалог Prefs, использовать его особого смысла нет.



› Видите стрелку острием вниз в левом верхнем углу? Это для подъема вверх по дереву каталогов. Интуитивно, да?



Rox-filer

Инструмент RISC OS формирует облик Linux.

Rox-filer – часть проекта *Rox Project*, намеренного принести некоторые функции RISC OS на настольные системы Unix. RISC OS стала популярна в Соединенном Королевстве благодаря машинам Acorn Archimedes, установленным в школах, и все еще разрабатывается маленькой, но преданной армией поклонников, ради ее скорости и дружественного пользовательского дизайна. Одна из особенностей дизайна – 'AppDir', где все программы упакованы в свои каталоги, а желаемое приложение можно запустить двойным щелчком по нему.

Файловый менеджер отказался от традиционной организации окна в пользу единой панели кнопок и иконок – в программе нет ни меню, ни дополнительных оконных элементов. Как и в RISC OS, все операции над файлами проводятся через контекстное меню, вызываемое правой кнопкой мыши. Поддерживается режим перетаскивания, а также отображение галерей файлов и режим «больших иконок» (нажмите на иконку увеличительного стекла со знаком «плюс» на панели кнопок). Можно задать подробный вывод информации о файлах, включающей размер, дату модификации и прочее.

К работе *Rox-filer*, настроенного по умолчанию, надо притерпеться: окно постоянно меняет размер, чтобы вместить все файлы, и размер иконок файлов зависит от числа файлов в каталоге – немного необычно. Кроме того, входят в каталог по однократному щелчку мыши. К счастью, поведение программы, как и множество косметических аспектов, можно изменить с помощью исчерпывающего диалога настроек.

Типы файлов в *Rox-filer* обрабатываются не блестяще. Текстовые файлы открываются в редакторе *MC*, а графические файлы просто выдают сообщение «неизвестно, что делать». Привязать к типам файлов обработчики не сложно, но стиль работы RISC OS предполагает, что для открытия файла вы перетащите его в программу. В целом, *Rox-filer* – славный небольшой файловый менеджер, пусть и с причудами; четкий интерфейс и простая навигация очень подойдут новичкам.

«Четкий интерфейс и простая навигация подойдут новичкам.»

В главном окне нет панелей инструментов – по традиции RISC OS, все делается через контекстные меню.



EmelFM2

Дизайн старой школы и прелесть *GTK2*.

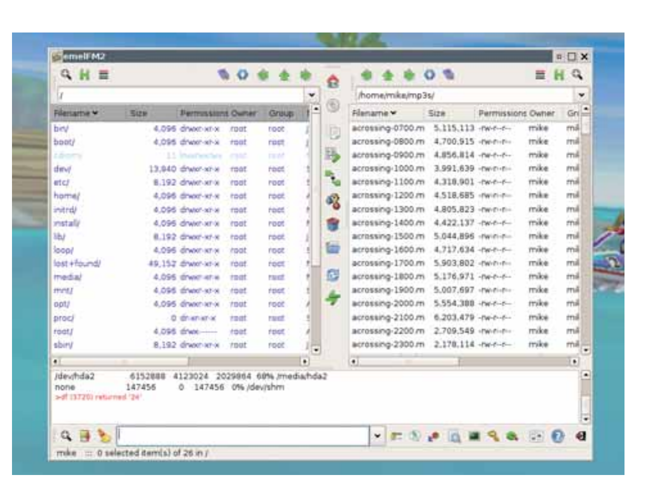
Как в *Gentoo* и *Midnight Commander*, в *EmelFM2* используется двухпанельный интерфейс, который по сердцу многим хакерам Unix. Но в игре против *Gentoo* у него есть козырь: он использует *GTK2*, поэтому лучше вписывается в темы и стили современных рабочих столов Gnome или *Xfce*. И потом, здесь нет ничего лишнего: текст на экране – только список файлов, остальное – кнопки для основных файловых операций.

Две панели с файлами независимы, и вы можете отобрать определенные типы файлов или перейти к закладкам на одной панели, не влияя на работу другой. «Нейтральная полоса» кнопок, разделяющая эти панели, позволяет вам копировать, перемещать или создавать символьные ссылки на файлы между панелями; с помощью контекстного меню, вызываемого правой кнопкой мыши, можно получить доступ к дополнительным функциям, например, разбиению файлов или сжатию их в различные форматы (gzip, bzip2, rar и т.д.).

Как и *MC*, *EmelFM* включает окно командной строки внизу экрана – оно принимает команды интерпретатора, но имеет несколько путаный вывод, поэтому лучше бы в качестве мини-терминала использовался *VTE*. Великолепна система расширений – пока что большинство из них работают как демонстрации, но в будущем позволят увеличивать функциональность программы, не раздувая ее основной код.

EmelFM приятно использовать: он быстр, не требует много оперативной памяти и прекрасно выглядит благодаря *GTK2*. Однако, у него есть большая проблема: нестабильность. В версии 0.2 вряд ли можно винить разработчиков за нередкие падения программы, с которыми мы столкнулись. Но их наличие препятствует широкому применению *EmelFM*. Если программа станет надежной к выпуску версии 1.0, ее можно рекомендовать как лучший менеджер файлов для настольных систем.

«Нередкие падения препятствуют широкому применению.»



Интерфейс *EmelFM2* – лучшее воплощение стиля *Commander*, встречавшееся нам в мире Linux.



Файловые менеджеры Вердикт

Konqueror 9/10

Самым большим открытием этого **Сравнения** стала колоссальная разница в работе файловых менеджеров. Как упоминалось вначале, все они могли казаться вам на одно лицо: нажмите здесь, сделайте это, попробуйте то. Но помочь пользователю управиться с необходимыми операциями, не удушив его графическими примочками – это род искусства, и не всем такая задача по плечу. Конечно, некоторые тестируемые менеджеры делают ставку на быстроту, пренебрегая удобством. Однако в борьбе Linux за рабочие столы пользователей иметь классный файловый менеджер принципиально важно.

И мы такой уже имеем. В *Konqueror* есть все: гибкость в применении, функции для продвинутых пользователей, а также возможность упрощения, чтобы и Linux-дебютанты тоже могли с ним справиться. Он не идеален, некоторые его функции можно улучшить (например, диалог настройки), но в целом работает прекрасно.

Nautilus отстал ненамного: он соответствует философии простоты Gnome, и хотя требовательные пользователи могут счесть его мощь недостаточной, обычным пользователям он предоставляет все необходимое.

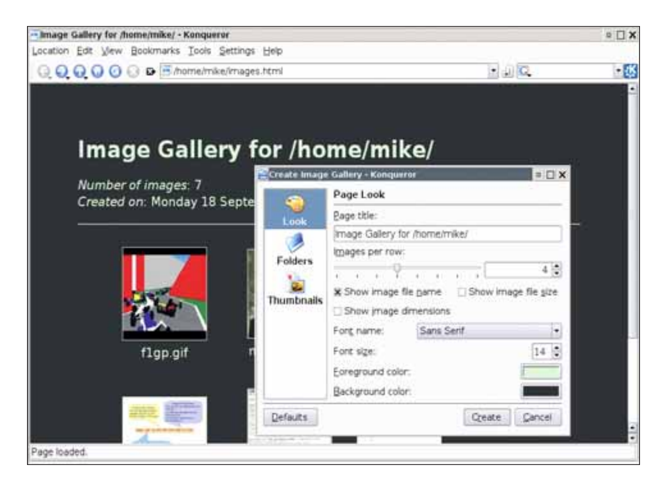
Красавец-урод

А как с остальными? *Gentoo* просто уродлив, он выглядит непостижимым для тех, кто никогда не использовал подобные файловые менеджеры, но опытные пользователи будут счастливы. Возможности его настройки настолько огромны, что его даже трудно назвать «программой» – при желании из него можно сделать почти все, что угодно! Основным недостатком Gentoo является время – много времени – чтобы настроить его по вашему вкусу, но единожды проделав эту работу, вы уже не захотите переходить к чему-нибудь другому. Если вам требуется что-нибудь помиловиднее, можно попробовать *EmelFM2*, но мы бы не стали его использовать с важными данными, пока не будет приведена в порядок стабильность.

Для легковесного рабочего стола, то есть старых ПК с *Fluxbox* или *IceWM*, можно рекомендовать *Rox-filer*. Дизайн приемлем для новичков, программа сохраняет изящество RISC OS – простоту и элегантность.

Nao или *Xfm* не рекомендуются для использования в сегодняшнем мире, разве что на престарелых машинах эпохи 80486: в противном случае вы лишь погрязнете в их трясине. Наконец, *Midnight Commander* вряд ли обоснуется на рабочих столах, но администраторам сервера стоит иметь его под рукой: уж очень хорошо он работает на медленных SSH-соединениях.

Теперь перейдем к фактам и цифрам... LXF



› *Konqueror* набит функциями: это генератор web-галереи, он сканирует картинки и создает HTML-страницу.

> «В борьбе Linux за рабочие столы важен файловый менеджер.»

Обратная связь

Не согласны с нашими оценками? Вы считаете, что некий файловый менеджер лучше всех перечисленных здесь? Хотите узнать, что говорил Эффи, когда их тестировал? Сообщите нам по почте letters@linuxformat.ru или зайдите на наш форум: www.liunuxforum.ru.

Сравнительная таблица

	EmelFM2	Gentoo	Konqueror	MC	Nao	Nautilus	ROX-Filer	Xfm
Лицензия	GPL	GPL	GPL	GPL	GPL	GPL	GPL	GPL
Интерфейс	GTK 2	GTK 1	Qt	Text	Fox	GTK 2	GTK 2	Xaw
Размер двоичного кода [1]	493K	448K	2.8K [10]	643K	714K	1.1MB	450K	175K
Время запуска	Быстро	Быстро	Медленно	Очень Быстро	Быстро	Медленно	Быстро	Очень Быстро
Скорость отображения	Быстро	Нормально	Медленно	Очень Быстро	Нормально	Нормально	Быстро	Очень Быстро
Требуемая память [2]	7MB	6MB	25MB	2MB	6MB	30MB	10MB	3MB
Назначение иконок [3]	☒	☒	☒	Не доступно	☒	✓	✓	☒
Предпросмотр файлов [4]	☒	☒	✓	Не доступно	✓	✓	✓	☒
Режимы отображения [5]	☒	☒	✓	Не доступно	✓	✓	✓	✓
Время, потраченное Эффи на переименование файла (сек) [7]	42	32	17	204	24	18	21	22
Время, потраченное Эффи на создание каталога (сек) [8]	41	54	60	12	17	10	26	21
Время, потраченное Эффи на смену режима просмотра (сек) [9]	Не доступно	Не доступно	15	Не доступно	8	11	7	29

[1] Размер исполняемого кода программы, [2] Отображение каталога из 1 100 файлов, [3] Один и тот же каталог для всех программ, [4] Можно ли назначать отдельную иконку каждому файлу? [5] Показывает ли файловый менеджер содержимое файлов, например, эскизы картинок? [6] Позволяет ли программа выбирать между иконками, списком и так далее? [7] В секундах, [8] См. пункт 7, [9] См. пункт 7, но меняли режим просмотра с иконок на список. [10] Размер кода невелик, потому что Konqueror на самом деле собирается из разделяемых библиотек KDE.

Distrowatch



Ежемесячная сводка новостей о дистрибутивах Linux.



ЛАДИСЛАВ БОДНАР
основатель, редактор, начальник и сотрудник DistroWatch.com.

APT-выбор

Видимо, главное отличие одного дистрибутива от другого – это система управления пакетами. *Urpmi* или *Yum*, *apt-get* или *Pkgadd*, *Emerge* или один из графических интерфейсов для этих инструментов – имеющийся менеджер пакетов часто обуславливает предпочтение пользователя. Так какой же из них наилучший?

После многолетнего опыта использования различных дистрибутивов и менеджеров пакетов, я убежден, что Debian GNU/Linux в этом плане почти идеален. *APT* (Advanced Package Tool – Продвинутый Инструмент для Пакетов) с самого начала спроектирован с тщательной проработкой деталей и с целью облегчить непрерывные обновления на любом уровне. Он быстрее всех соперников, и теперь, с представлением *APT 0.6*, обзавелся функциями безопасности, а именно проверкой цифровой подписи и криптографией.

Не столь Yum'ильны

Напротив, *Yum* от Fedora выглядит медленным и неуклюжим, а *You* в OpenSUSE лишь недавно позволил включать в конфигурацию сторонние репозитории. *Urpmi* в Mandriva тормозит при обновлении базы данных пакетов и склонен к зависанию, если зеркало, с которого производится обновление, лишь частично синхронизировано с главным сервером. Инструменты управления пакетами в Slackware примитивны (к примеру, не устанавливают зависимости), а *Emerge* в Gentoo, сам по себе превосходный, применяется лишь любителями частой компиляции программ.

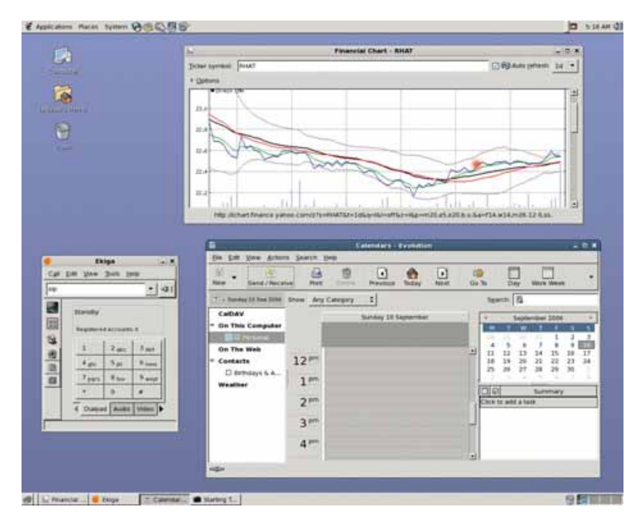
Несомненно, у вас свой опыт работы с различными утилитами управления пакетами в Linux, так что жду ваших мыслей...

ladislav.bodnar@futurenet.co.uk

Красный сигнал

Red Hat Enterprise Linux 5 Дистрибутив для предприятий сулит средства мультимедиа, новую клиент-серверную структуру и виртуализацию *Xen*.

С выходом первой бета-версии Red Hat Enterprise Linux (RHEL) 5 в начале сентября, пользователи и бета-тестеры известнейшего в мире дистрибутива Linux уровня предприятия получили пару сюрпризов. Прошли те дни, когда этого продукт считался просто множеством «продвинутых» (AS) и «корпоративных» (ES) серверов и рабочих станций (WS). Теперь RHEL 5 разбила свой базовый код на «клиентскую» и «серверную» части – это более интуитивно-понятный способ обозначения целей каждого продукта. Дополнительную ясность вносит тот факт, что клиентское издание доступно только для архитектур i386 и x86_64, а серверное предоставляется также для Itanium2 от Intel, eServer и процессоров Power от IBM.



› **Клиентская редакция RHEL 5 поставляется с последними версиями финансового, офисного ПО и средств для совместной работы.**

ПО поставляется в «медиа-пакетах» (media-kits): группах программных пакетов, разработанных для конкретной функциональности. Со стороны сервера, вы найдете набор средств для организации кластеров и балансировки нагрузки web, получите доступ к параллельным устройствам хранения и виртуализацию *Xen*. В клиентской части фигурируют рабочий стол (с Evolution и OpenOffice.org), рабочая станция (с полным набором средств разработки) и... опять виртуализация *Xen*. Пользователь выбирает набор инструментов при установке.

> «Stateless Linux — новый способ управления большими системами.»

Кстати, о *Xen*: это не просто модное словцо из технологических публикаций, но также концепция, энергично продвигаемая Red Hat и Novell, двумя лидерами корпоративных Linux-решений, и они правы. Идея запуска различных ОС (или нескольких копий одной и той же ОС) одновременно на одной машине, похоже, формирует будущее бизнес-компьютеризации. *Xen* исключительно полезен для хостинг-провайдеров, так как помогает им предоставлять виртуальные выделенные сервера. Другим фирмам, использующим службы высокой доступности, тоже пригодятся преимущества этой технологии. *Xen 3.0* включен в RHEL 5.

Xen проталкивают уже давно, но у RHEL 5 есть тузы в рукаве и поинтереснее – например, концепция Stateless Linux как удобный способ управления большим количеством Linux-систем: системный администратор может настраивать сотни Linux-клиентов в сети и поддерживать их синхронизацию с базовой системой, то есть, по сути, создавать заменяемые «киоски». Среди других функций – интеграция смарт-карт и представление InfiniBand и Remote Direct Memory Access для сетей с высокой пропускной способностью и низким временем ожидания.

Хотя Red Hat все еще очевидный лидер рынка корпоративного Linux, Novell бросает им вызов своим последним выпуском SUSE Linux Enterprise Server и Desktop 10. Новые продукты Novell, с уже интегрированной технологией *Xen*, весьма положительно оценены в прессе. Но с учетом новых технологий, включенных в RHEL 5, похоже, что Red Hat вновь обскакала Novell.

Финальный релиз Red Hat Enterprise Linux 5 должен выйти в начале 2007 года.

www.redhat.com

Лавка вкусностей

Mandriva Linux 2007 Популярная ОС вернулась с Live-опцией и мощной поддержкой 3D.

Сейчас, когда вы это читаете, Mandriva Linux 2007 уже должна быть доступна для продажи и загрузки через ее клубный сервис. Судя по первым релиз-кандидатам, продукт сделан с прицелом отвоевать большой кусок на арене дистрибутивов Linux. Сделан упор на внешний вид, а 3D-эффекты, новейшие пакеты и множество вариантов установки, а также Live CD (последнее предложение от парижского изготовителя дистрибутивов), похоже, привлекут много новых пользователей, особенно после выхода свободно-загружаемой версии.

Большая новость версии 2007 – беспрецедентная поддержка 3D-эффектов рабочего стола: она включает и *AIGLX* (разработка, поддерживаемая Red Hat), и *Xgl/Compiz* (технология, созданная Novell). Некоторые из дисков Mandriva One Live CD включают проприетарные графические драйвера от ATI и Nvidia, так что, запустив Live CD, пользователи смогут насладиться необычным трехмерным рабочим столом.



› **Графический инструмент для настройки 3D-эффектов – это главное.**

В доказательство заботы не только о внешней привлекательности, Mandriva 2007 предоставляет DVD, поддерживающий архитектуры i586 и x86_64. Любой пользователь, работающий с конгломератом серверов и настольных компьютеров на различных процессорах, высоко оценит удобство обладания всего одним установочным носителем для всех своих компьютеров. Кроме DVD, выпущено множество установочных Live CD с рабочими столами Gnome и KDE для различных языков. Коммерческие издания Discovery и PowerPack предназначены для, соответственно, начинающих и опытных пользователей.

www.mandriva.com

› Mandriva будет на диске-приложении в следующем месяце.

Потомок Slack'a

Zenwalk Linux 3.0 Небольшой дистрибутив на базе Slackware для мультимедиа и разработки.

Zenwalk Linux – дистрибутив скромный, но он регулярно получает хорошие отзывы в форумах сообщества пользователей. Основанный на Slackware Linux, но без KDE и других больших программных пакетов (отсюда и его прежнее имя – Minislack), Zenwalk тяготеет к среднему пользователю Linux, кому нужен быстрый и нетяжелый рабочий стол, хорошая поддержка мультимедиа «из коробки» и полный набор необходимого ПО и библиотек для разработки.

Стабильные релизы Zenwalk Linux выходят примерно раз в два месяца и помогают быть в курсе того, что творится в мире Open Source.

Разработчики создали мастер конфигурации на базе *Ncurses*, выскакивающий сразу при первой загрузке. В нем можно создать одну или более обычных (не-root) учетных записей пользователя, настроить сеть и X-сервер и даже активировать клавишу Num Lock. По завершении этих шагов система будет готова к работе.

Главный козырь Zenwalk – наличие достойной утилиты управления пакетами (т.е. она и вправду отслеживает и



› **Жан-Филипп Гийемен – основатель Zenwalk Linux.**

разрешает зависимости). Называется она *Netpkg*, это удобный инструмент для установки и обновления программ через сеть.

Благодаря отличному внешнему виду, превосходному составу приложений, удобным утилитам конфигурирования и управления пакетами и малому релиз-циклу, Zenwalk Linux быстро становится стандартом рабочего стола среди последователей Slackware. Попробуйте его! LXF

www.zenwalk.org

Где корни вашего дистрибутива?

Многие разработчики дистрибутивов создают новые продукты, взяв за основу один из существующих «основных» дистрибутивов. Исторически такой основой служил Red Hat Linux, но позднее, особенно после появления Knoppix и концепции Live CD, для частных решений стал шире применяться Debian. Slackware Linux тоже набирает популярность среди разработчиков, но Mandriva и особенно OpenSUSE, как ни странно, выбираются редко. Ниже приведен список основных дистрибутивов с их потомками. Замечание: данная таблица может утратить точность ко времени прочтения вами, ведь из-за стремительной скорости разработки новые дистрибутивы создаются и исчезают практически каждый день.





Дистрибутив	Потомков	Примеры
Debian GNU/Linux	124	Ubuntu, SimplyMepis, Knoppix
Fedora Core	51	Aurox, Fox Desktop, Yellow Dog
Slackware Linux	46	Slax, VectorLinux, Zenwalk
Red Hat Enterprise Linux	21	Asianux, CentOS, Red Flag
Gentoo Linux	18	Kororaa, Ututo, VLOS
Mandriva Linux	12	MCN Live CD, PCLinuxOS
FreeBSD	11	DesktopBSD, FreeSBIE, PC-BSD
OpenSUSE	4	Linux Caixa M gica, StressLinux

Хит-парад

10 самых посещаемых страниц на DistroWatch.com с 15 августа по 14 сентября 2006 (среднее число визитов в день)

Дистрибутив	Количество визитов	
1 **Ubuntu**	2,375	‹›
2 **SUSE**	1,944	‹›
3 **Fedora Core**	1,199	↑
4 **PCLinuxOS**	1,075	↓
5 **SimplyMepis**	1,058	‹›
6 **Mandriva**	937	‹›
7 **Slackware**	918	‹›
8 **Debian GNU/Linux**	826	↓
9 **Damn Small Linux**	737	↓
10 **Gentoo**	728	↑

» DistroWatch.com отслеживает популярность дистрибутивов, основываясь на количестве посещений сайтов, посвященных конкретным дистрибутивам. Хотя эти цифры и не отражают реальное количество инсталляций, они являются индикатором популярности дистрибутива на данный момент времени.



Many happy return()s! Долгих лет тебе, Linux!

За время, прошедшее с 1991 года студенческий проект вырос в свободную операционную систему, на которую рассчитывают и которой наслаждаются предприятия и пользователи во всем мире. **Нейл Ботвик** и **Энди Ченнел** разыскали хакеров, которые помогли этому случиться.

«Я пишу (бесплатную) операционную систему (это просто хобби, она не будет такой большой и профессиональной, как gnu) для клонов AT 386(486)»

«Просто хобби»? Только для клонов 386? Разве это наводит на мысль об ОС, способной перевернуть мир? Однако именно так Линус Торвальдс объявил о надвигающемся рождении Linux 15 лет назад. С того раннего и неамбициозного начинания Linux перерос самые смелые мечты своего создателя. Работая на суперкомпьютерах, мобильных телефонах и на множестве устройств между ними, GNU/Linux теперь существует во многих формах и дистрибутивах и продолжает развиваться. Он обеспечивает прекрасную платформу для движения Open Source и предлагает свободную, быструю и безопасную работу на компьютере пользователям всего мира. На следующих нескольких страницах мы отметим его 15-й день рождения, оглядываясь на его подъемы и спады. Мы поговорим с людьми, вовлеченными в разработку Linux, чтобы разобраться, как он возник, где и почему используется.

Мы также смахнем пыль с нашего (открытого и прозрачного) хрустального шара и спросим, чего ожидать от Linux, когда он перейдет из подросткового возраста к полной зрелости. Здесь применима обычная оговорка: все прогнозы – это личные мнения, и вам не следует вкладывать деньги в акции Linux-компаний, основываясь лишь на наших предположениях (но вспомните о нас, пожалуйста, если поймаете свою удачу). Однако начнем с истоков...

На этих страницах

Что сталось с...

Питером МакДональдом?
МакДональд создал Softlanding Linux System (SLS), ставшую источником вдохновения для авторов Slackware, Debian и SUSE. Теперь он владеет фирмой PDQ Interfaces в Британской Колумбии (Канада), занимающейся разработкой ПО и консалтингом (http://pdqi.com/pdqi). Подробности на стр. ??.

1991–1992 Младые годы

Скромные истоки истории Linux.

Оглянувшись назад, мы часто можем точно указать разговор или событие, перевернувшее нашу жизнь: начало новой карьеры или новых взаимоотношений, или конец старых. В случае с молодым финским студентом компьютерных наук это было сообщение в Usenet, которое он разместил в группе новостей comp.os.minix 25 августа 1991 года.

Эта дата считается днем рождения Linux. Фактически, первый релиз Linux, под номером 0.01, вышел спустя несколько недель; версия 0.02 последовала в начале октября. Linux пересек отметку 0.10 в декабре, менее чем через четыре месяца после первоначального анонса. Вот сообщение, положившее начало всему:

> Привет всем, использующим minix – я делаю (бесплатную) операционную систему (это просто хобби, она не будет такой большой и профессиональной, как gnu) для клонов AT 386(486). Я занимаюсь ею с апреля, и кое-что уже вырисовывается. Буду рад любым отзывам о том, что народу нравится/не

Ключевые даты

Январь 1991. 21-летний студент Линус Торвальдс (L), изучающий компьютерные науки в Университете г. Хельсинки, покупает ПК 386 с 33 МГц-процессором, чтобы играть в *Prince of Persia*, и начинает писать Unix-подобную операционную систему для 386, используя книги Энди Таненбаума и Мориса Баха.



Программировать он научился на папином Vic-20.

Июнь 1991. Ричард Столлмен (S) публикует вторую версию своей сотрясающей основы GNU General Public License, которая разрешает пользователям брать чужой код, коль скоро они выпускают плоды своих трудов под той же лицензией. Считается, что логотип проекта, голову антилопы-гну (G), нарисовал Этьен Сувиса.

Август 1991. Торвальдс на comp.os.minix сообщает миру, что пишет некую ОС, но она не будет «большой и профессиональной, как GNU». Рабочее название – Freax.



Сентябрь 1991. Первая версия (0.01) того, что теперь называется Linux, выпущена с аппаратной

нравится в minix, поскольку моя ОС кое в чем похожа на нее (помимо всего прочего, у нее тот же самый физический уровень файловой системы (по практическим соображениям)). На данный момент я портировал bash(1.08) и gcc(1.40), и похоже, что они работают. Значит, в течение нескольких месяцев получится нечто практически полезное, и хотелось бы знать, какие функции нужны большинству. Приветствуются любые предложения, но я не обещаю, что реализую их :-). Линус (torvalds@kruuna.helsinki.fi). P.S. Да — она свободна от какого-либо кода minix и имеет многопоточную файловую систему. Она не переносима (использует переключение задач 386 и т.д.) и, вероятно, никогда не будет поддерживать что-то кроме жестких дисков AT, потому что это все, что у меня есть :-(.

Как подсказывает это сообщение, Торвальдс работал на операционной системе Minix, написанной профессором Эндрю Таненбаумом [Andrew Tanenbaum] в конце восьмидесятых. Позже Таненбаум принял на вооружение дружелюбного зверька-талисмана (он выбрал енота), как и Linux. Таненбаум использовал Minix для обучения студентов операционным системам, но он не смог выпустить исходный код Minix как поистине свободно распространяемую ОС. С другой стороны, быстрые темпы развития Linux, как и сейчас, стали возможными не только благодаря программистским усилиям Торвальдса (он все еще был студентом-очником Хельсинкского университета), но также благодаря его предусмотрительности в выпуске исходного кода с самого начала. «Я хотел, чтобы он был легко доступен по FTP с полными исходными текстами, и не хотел, чтобы это для кого-то было слишком дорого», говорит он. Как только он сделал это, программисты-энтузиасты начали скачивать и подправлять ядро, возвращая свои дополнения Торвальдсу для включения в следующий релиз.

На этом этапе ОС называлась Freax, игра слов «free» (свободный), «freak» (чудик) и «x», обычно используемая в наименовании Unix-подобных систем. Название Linux непредумышленно создал Ари Лемке [Ari Lemmke], администратор FTP-сервера, где размещался код. Он назвал каталог **linux**, и Торвальдс решил, что пусть так и будет. (Правильное произношение слова «Linux» – с коротким i, а не с длинным. Если есть сомнения, в разделе **Magazine/Linus** на DVD приведен пример того, как этот человек сам демонстрирует правильное произношение).

Вхождение в GNU

Для загрузки первоначальных релизов новой ОС все еще требовалось наличие на компьютере другой ОС, но это изменилось с приходом загрузчика *Lilo* (LInux LOader), написанным шведским ученым Вернером Алмесбергером [Werner Almesberger] в 1992 г. Исключение необходимости использовать другую ОС стало шагом к независимости Linux, но основная функция любой ОС – запуск программ. Без программ, запускаемых под Linux, новая ОС была бы просто курьезным упражнением по информатике. Случилось так, что Фонд свободного ПО (Free Software Foundation) разрабатывал свою собственную ОС GNU (Gnu is Not Unix) и ПО, начиная с 1984 года. ОС была почти готова, кроме одного важного компонента: ядра. Решение было очевидно, за исключением того, что новое ядро Linux тогда еще не распространялось под GPL. В начале 1992 года Linux должным образом вышел под лицензией GPL, и началась работа по объединению Linux и программ GNU. Вот почему многие, и совершенно справедливо, настаивают, что Linux следует называть GNU/Linux.

GNU и Linux были и остаются отдельными проектами, но они подпитывают друг друга. Linux был бы гораздо менее полезен без инструментария GNU, работающего на нем, в частности, без компилятора GCC, используемого для сборки практически любой программы на вашем компьютере. «Выпуск Linux под GPL был, определенно, лучшей вещью, которую я когда-либо делал», – рассказывал Торвальдс Хироо Ямагате [Hiroo Yamagata] в 1997 г.

GNU, в свою очередь, получила ядро для запуска своих программ (собственное ядро GNU, Hurd, появится незадолго до того, как Солнце станет красным гигантом) и огромную массу разработчиков, благодаря популярности Linux. »

Интервью: проф. Эндрю Таненбаум

Когда Линус Торвальдс решил писать собственную ОС, он обратился к конструктивной книге Эндрю Таненбаума, посвященной *Minix*. Таненбаум, известный как «ast», сейчас – профессор информатики Университета Врейе в Амстердаме и все еще активное участвует в разработке операционных систем.



LXF: Когда вы писали книгу о Minix, вы ожидали, что она окажет такое влияние за пределами академии?

ЭТ: Нет, я ожидал, что нишу свободных ОС займет BSD, или, возможно, GNU. BSD погрязла в судебной тяжбе с AT&T, которая не пускала ее на рынок несколько очень важных лет, а GNU даже 15 лет спустя так и не появилась. Эти два «несчастных случая» дали Linux пространство для расширения. Будь BSD во всем цвете доступна в 1993 году, с тысячами приложений, многолетним тестированием и распространенностью по всему миру, Linux бы не жить.

LXF: А вы знали о Linux как о производной от Minix в годы его становления?

ЭТ: Некоторое время я следил за ранней разработкой Linux. Вспомните, что была большая группа новостей comp.os.minix (да и сейчас она существует), и что Линус работал на Minix несколько лет до написания Linux. Кстати, Minix 3 все еще живехонька. Примерно 90 тысяч человек скачали ее в прошлом году.

LXF: Была ли частью раннего успеха Linux апелляция к желанию студентов и профессионалов возиться с ней?

ЭТ: Не преувеличивайте влияние тысяч фанатов Linux, Линус писал ядро и руководил им сам. Масса основных улучшений исходят от горстки программистов, нанятых и оплачиваемых парой фирм типа IBM и Red Hat. Однако пользователи внесли свой вклад в разработку или портирование множества приложений и драйверов.

LXF: Вы рассматриваете возможность модифицировать Linux как неотъемлемую сильную сторону, или это могло бы привести к множеству клонов Linux?

ЭТ: Взгляните на то, что произошло с BSD. Она разделилась на три или четыре ветви, которые по большей части борются друг с другом. В результате они намного слабее, чем могли бы быть как один единственный дистрибутив. Пока что Linux избегал этой судьбы, во многом благодаря способности Линуса направлять всех в одну сторону.

Знаете ли вы?

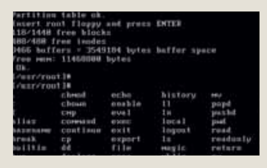
В 1994 году Йон 'Мэддог' Холл из Digital Equipment Corporation убедил Линуса портировать Linux на 64-разрядный процессор DEC Alpha. Он также уговорил DEC передать Линусу рабочую станцию Alpha. Холл полагал, что «DEC и сообщество Linux сформировали первое подлинно успешное совместное предприятие «пиджаков» и «чокнутых линуксоидов».

Мы глубоко признательны Red Hat за этот древний снимок коробки с Linux для Alpha

поддержкой для финских клавиатур.

Декабрь 1991. К версии 0.11 проекта Торвальдса в ОС добавлена поддержка гибких дисков. Торвальдс теперь рассматривает ее как самостоятельную систему, независимую от Minix.

Декабрь 1991. Эндрю Триджелл, австралийский аспирант, приступает к решению задачи получения доступа к разделяемым ресурсам сервера, используя протокол Microsoft SMB (Server Message Blocks). Первый релиз выходит пару месяцев спустя. Триджелл забросит этот проект, но перейдет в Linux в ноябре 1992 года и позже использует SMB как основу для Samba.



Февраль 1992. Проект GNU принимает выпущенный к тому времени под GPL Linux в качестве своего ядра, как «затычку», до завершения Hurd. Столлмен начинает свою эпопею, пытаясь заставить каждого говорить «ГНУ-слэш-Линукс», а не просто «Линукс».

Март 1992. Орест Зборовски (Orest Zborowski) берет Linux за основу для своей платформы X386 – вскоре переименованной в XFree86. *X Window* начинает работать на Linux 0.96 в апреле.



1993—1996 Первые ласточки

Как ядро и несколько свободных программ стали дистрибутивами.



В наши дни слово «Linux» (или, возможно, «GNU/Linux») используется для ссылки на завершенную совокупность ОС и ПО, но так было не всегда. Linux был вначале доступен только как ядро: вы ставили его, затем добывали другое ПО, нужное для сборки работающей системы. Решением стало связать все это в один пакет для установки и распространять его.

Спорят о том, какой Linux-дистрибутив был первым. Slackware Патрика Фолькердинга [Patrick Volkerding] принято считать старейшим из ныне здравствующих дистрибутивов, но многие называют первым Yggdrasil. Проектом руководил Адам Ритчер [Adam Ritcher], специалист по *X Window* с ученой степенью в области компьютерных наук в Калифорнийском Университете (Беркли). Дебютировав в феврале 1993 года, Yggdrasil стал первым дистрибутивом, выпущенным на CD-ROM и реализующим некоторые продвинутые концепции, например, распознавание Plug-and-Play устройств и вариант LiveCD – то, что мы сейчас воспринимаем как само собой разумеющееся. «Помнится, я поставил версию Yggdrasil Linux и следил за загрузкой *X Window* и компиляцией Samba в окне *xterm*», говорит Джереми Эллисон [Jeremy Allison]. «Я решил, что переведу все мои рабочие станции Sun на Linux... Через несколько лет он стал моей единственной настольной платформой».

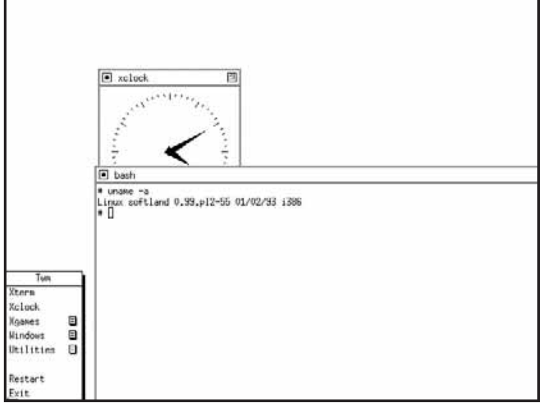
Еще один из первых дистрибутивов назывался Softlanding Linux System – его раннюю версию вы найдете на нашем диске. Подобно многим Linux-хакерам, его автор Питер МакДональд [Peter MacDonald] увлекся Linux в университете. «Сначала – разрабатывая заплатки к ядру, затем – собирая воедино и пытаясь поддерживать дистрибутив», вспоминает он. Детище МакДональда основывалось на ОС под названием MCC Linux, которую разрабатывал Массачусетский компьютерный центр с 1992 г. MCC Linux безнадежно отставал от столь функциональных дистрибутивов, как Yggdrasil, и просуществовал недолго, но его наследие в качестве основы для Softlanding Linux System очень важно, поскольку SLS, в свою очередь, стал отправной точкой и для Debian, и для Slackware.

Рождение старой гвардии

Итак, к середине 1993 года разработка дистрибутивов стремительно разрасталась, и технологии распространения на CD-ROM, поддержка оборудования и графика придвигались к своим пределам усилиями сообщества студентов и программистов – приверженцев Linux, общавшихся через Usenet. «Я помню, что было вперемешку много и удовольствия, и работы», говорит МакДональд. «Было множество заманчивых троп, но в то же время беспокоила разобщенность и раздробленность».

В августе 1993 года Ян Мердок [Ian Murdock] объявил о «грядущем завершении» нового дистрибутива, названного Debian Linux Release. Хотя Мердок начинал с изменений в SLS, он был все больше и больше недоволен им, и решил основать собственный дистрибутив с нуля. Если Yggdrasil получил свое имя из норвежской мифологии, название Debian было составлено из имени подружки (ныне жены) Мердока – Дебры [Debra] и его собственного (Ian); о произношении тут спорить не приходится. В анонсе выпуска были подробности о том, что Debian будет содержать и делать, включая «Debian будет содержать все почти самое современное. Систему будет легко поддерживать в актуальном состоянии с помощью сценария обновления в базовой системе, который будет обеспечивать полную интеграцию пакетов обновлений». Хотя вы можете хихикнуть над первым предложением, простота обновлений – определенно одна из сильных сторон Debian.

Дистрибутив достиг практичной стадии в начале 1994 года с выпуском версии 0.91, и прежде всего выделялся системой управления пакетами. Мердок также написал Манифест Debian, документ, разъясняющий причины и цели Debian, включая его приверженность к свободному ПО. Debian был, да и остается, проектом сообщества. Под руководством Мердока Debian устойчиво рос и начал поддерживать платформы, отличные от i386, на которое первоначально рассчитывал Торвальдс. Это до сих пор остается в центре внимания проекта – текущий стабильный релиз работает на 11 различных архитектурах – и приводится как одна из причин осторож-



› **Softlanding Linux System стал стартовой площадкой для Патрика Фолькердинга, Яна Мердока и многих других хакеров первой волны.**

Что сталось с...

Документом Linux FAQ?



В 1992 году, до появления «всемирной паутины», Linux-документация в основном была доступна в одном файле: *Linux FAQ*. Инициатива стала перерастать в массу документации, которую разработчики ядра и администраторы едва могли распечатать. Linux Documentation Project, как он стал называться, разросся в коллекцию руководств и HOWTO, и развился в один из самых полных, будь то Linux или что-то другое, ресурсов документации в Сети.

Ключевые даты

Август 1993. Ян Мердок (M) основал проект Debian, нацеленный на улучшение Softlanding Linux System и следующий духу GPL.



Январь – март 1994. Выпущены Debian 0.91 и Slackware 1.1.2. Марк Юинг основал Red Hat, выпустив v1.0.

Март 1994. Линус Торвальдс анонсировал Linux 1.0, с исходными текстами объемом 1 МБ. Первая заплатка для этой ОС появилась пару дней спустя.

Апрель – октябрь 1994. SUSE выпустила первую «бету» S.u.s.E Linux 4.2, пронумерованную, видимо, из уважения к смыслу жизни, вселенной и всему остальному. Торвальдс окончил Хельсинкский университет со степенью бакалавра. В Линдоне (штат Юта) Рэнсом Лав [Ransom Love] и Брайан Спаркс [Bryan Sparks] основали Caldera Systems, чтобы производить

ного цикла разработки. Можно смеяться над тихоходными релизами, но каждый администратор системы, работающей под управлением Debian Stable, может только кивнуть и тихо улыбнуться, зная, что на их компьютерах работает то же ПО, что и в прошлом месяце, и месяц до этого, и на пути их ожидает совсем немного сюрпризов.

Другой проект, основанный в 1993 г., принял совершенно другую этику, чем Debian. Это была Red Hat, фирма, основанная предприимчивым Марком Юингом [Marc Ewing] с целью производить лучший дистрибутив Linux. Red Hat предприняла свою первую попытку в следующем году. Red Hat Linux 0.9 был бета-версией, но на нем висел ценник. «Моей целью было получать достаточно денег, чтобы вести мой хакерский образ жизни – работать над Linux в своей скромной спальне», рассказывал Юинг для Salon.com в 1999 г. «Я рассчитал, что нужно продавать только 1500 копий каждый год – это мелочь! – и мне бы хватило на жизнь».

RHL 0.9 был, вероятно, первым дистрибутивом, снабженным графическим инсталлятором и графическими инструментами настройки. Они охватывали учетные записи пользователей и группы, **/etc/fstab**, время и дату, а также сеть. Последнее было, наверное, самым важным, поскольку это было, и в какой-то степени и сейчас остается, одной из наиболее сложных частей настройки Linux-систем.

Запахло деньгами

Ранние версии Red Hat использовали систему управления пакетами rpp. В 1995 г. вышел RHL 2.0, оснащенный новым Red Hat Package Manager, RPM. Бизнес был на взлете. Юинг объединил усилия с Бобом Янгом [Bob Young], который описал нам свои ярчайшие воспоминания о том времени: «На UNIX Expo в Нью-Йорке в сентябре 1995 года наша маленькая фирма Red Hat из кожи вон лезла, чтобы оплачивать свои счета. В первый день выставки в наш маленький бокс зашел джентльмен в синем костюме. Когда я спросил, что его интересует в Linux, он заявил, что это любопытно, но как директор по ИСУ в крупном нью-йоркском банке, он никогда бы не позволил своим системным администраторам где-либо применять Linux. Все же за последующие три дня выставки четыре системных администратора подошли к нашему стенду и купили по копии Red Hat Linux. Когда я спросил, как они будут использовать Linux на работе, передав мой разговор с руководителем их банка в первый день, все они отвечали что-то типа: «Начальство не дает нам должного финансирования, чтобы делать то, что они требуют, так что мы используем серверы Linux – просто им не говорим. Средств на замену не хватит, даже если их обнаружат».

RPM был перенят SUSE, когда они запустили свой дистрибутив S.u.S.E Linux 4.2 в 1996 году. Обратите внимание на маленькую «u» – в то время эта аббревиатура означала «Software und System Entwicklung» (разработка ПО и систем, – нем.). SUSE начала работать в конце 1992 года как консалтинговая Unix-организация, и производила пакеты программ на основе SLS и Slackware, но версия 4.2 стала переломным продуктом. Хотя она не происходила от Red Hat, но приняла ряд его особенностей, например, RPM и кое-что из структуры системы.

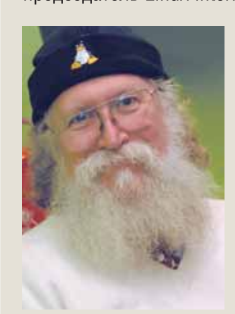
> «Свободное ПО тогда понимали даже меньше, чем сейчас.»

Red Hat и, в меньшей степени, SUSE, мгновенно повысили престиж Linux, и они, а не Debian или Slackware, стали самыми известными именами Linux за пределами сообщества, особенно среди бизнес-пользователей. Debian мог оставаться выбором энтузиастов и сторонников свободного ПО, но предприятия хотели заключать контракты на техническую поддержку и книги-руководства, чтобы обосновать свои расходы. Идея свободного ПО была еще менее понятна людям, чем сейчас, зацикливая большинство на трактовке слова «free» как «бесплатно». Предприятия с подозрением относились к «халявным» продуктам, так что коммерческий дистрибутив был необходим: не только для них и для Red Hat, но и для пользы всего Linux.

»

Интервью: Йон «Мэддог» Холл

Йон Холл – человек, благодаря которому Линус Торвальдс приложил руку к системе Alpha DEC. Холл говорит, что начал использовать свободное, открытое ПО еще в 1969 г. Сегодня он – председатель Linux International, некоммерческой организации.



LXF: Вы столкнулись с Linux и Линусом на раннем этапе. У вас были какие-то мысли насчет потенциала проекта?

ЙМХ: И да, и нет. Прежде всего я подумал о Linux как о проекте для образования и научных исследований. Это одна из причин, почему мне захотелось портировать его на Alpha. При выполнении исследований с проприетарной системой возникают трудности, когда нужно опубликовать ваше исследование. В случае свободного ПО вы можете просто сказать: «Вот код... работайте с ним и помогите мне сделать его лучше». Однако вскоре я начал видеть Linux в «реальных» проектах, и я думаю, системы *Beowulf* стали для меня первыми показателями коммерческой ценности Linux. Скорость, с которой продвигались проекты *Beowulf*, просто захватывала.

LXF: Какие факторы, по вашему мнению, позволили Linux процветать, в отличие от других ОС?

ЙМХ: На этот счет есть множество теорий... но может быть, просто он оказался нужной вещью в нужное время: резкое снижение цен на оборудование, повсеместное распространение Интернета, и этот вежливый молодой парень из Хельсинки с волосами песочного цвета, любитель пингвинов...

LXF: Насколько важным для Linux было вмешательство фирм вроде IBM и HP в период работы над ядром 2.4?

ЙМХ: Думаю, что крупные поставщики систем были очень важны по нескольким причинам. Они не только платили зарплату некоторым из наиболее активных разработчиков ядра, чтобы те могли продолжать делать свою работу все время, они также придали дух легитимности идее свободного ПО. Не думайте, что я считаю свободное ПО нелегитимным, но есть люди в мире, которые не поверят, что Земля круглая, пока крупная корпорация не скажет им об этом факте, и IBM (в частности) с ранних дней помогала ускорять рынок FOSS. Хорошие примеры их лидерства – это открытие IBM своего пула патентов для проектов FOSS, и IBM показывает, что услуги – это хорошая модель зарабатывания денег.

Знаете ли вы?

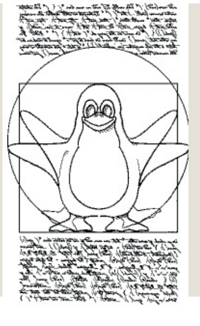
Эмблеме пингвина, выполненной Ларри Юингом, дал имя Джеймс Хьюджес [James Hughes], пояснивший, что это сокращение от «(T)orvalds (U)ni(X)» (именно так!). До чего же извилистый акроним пришлось изобрести – конечно, его назвали Tux [амер. смокинг, – прим. перев.] по причине, которая сразу бросается в глаза: он выглядит так, как будто носит смокинг. Вы можете лицезреть начальные изображения на www.isc.tamu.edu/~lewing/linux.

дистрибутив Caldera OpenLinux.

Апрель 1995.
Первый публичный релиз *Apache Web Server* (0.9.2). Он был построен на фундаменте HTTP Daemon Роба МакКула [Rob McCool] из Национального центра суперкомпьютерных приложений (NCSA).

Ноябрь 1995. Первый порт ОС на архитектуру Alpha. Порт «укрощался» Линусом на машине Alpha, которую раздобыл для него Йон 'Мэддог' Холл.

Январь 1996. Линус портирует ОС на архитектуру MIPS. Порт работает на машинах с R4x00, типа DECStation 5000, с поддержкой (в перспективе) более ранних машин.



Май 1996. Во время обсуждения вариантов талисмана для ОС, Торвальдс остановился на изображении дружелюбного пингвина, которое можно было легко анимировать, в отличие от неодушевленного логотипа Windows. Ларри Юинг [Larry Ewing] предложил дизайн.

Июнь 1996. Выпущен Linux 2.0. Исходный код раздулся до 5 МБ, и в **logo.gif** содержалось изображение пингвина Тукса [Tux].

1997—2001 Бум

И вдруг Linux появился повсюду... как и Интернет.



Период с 1997 по 2001 год был свидетелем сумасшедших дней бума дот-комов, когда под какую попало идею сделать деньги в Сети каждый мог получить невероятные инвестиции от венчурного капитала, по крайней мере, так казалось. Тогда Linux действительно начал расти. Разработчик ядра Алан Кокс [Alan Cox] вспоминает: «[Linux] начал расти как снежный ком где-то в 1996-м или около того. В 1995-м он был интересной технической загадкой, в 2000-м – большим бизнесом». Бум дот-комов продлился недолго, но он подтолкнул развитие Интернета, вывел его на первое место и помог расцвести Linux.

Связь между Linux и Интернетом – подлинный симбиоз. Рост числа интернет-соединений, и для домашнего использования, и для бизнеса, означал, что провайдерам и хостинговым компаниям нужно больше серверов: Linux на сравнительно дешевом оборудовании i386 был идеальным решением. Высвободившиеся деньги шли на финансирование новых проектов, позволяя большему числу оплачиваемых разработчиков работать над ключевыми открытыми проектами. В то же время, большее число людей в сети означало больше людей, интересующихся Linux, и значительно увеличивало число тех, кто мог содействовать движению, либо как полноценный разработчик, либо просто заполняя отчеты об ошибках и помогая тестировать ПО.

> «Топор войны был зарыт — но не умерло соперничество.»

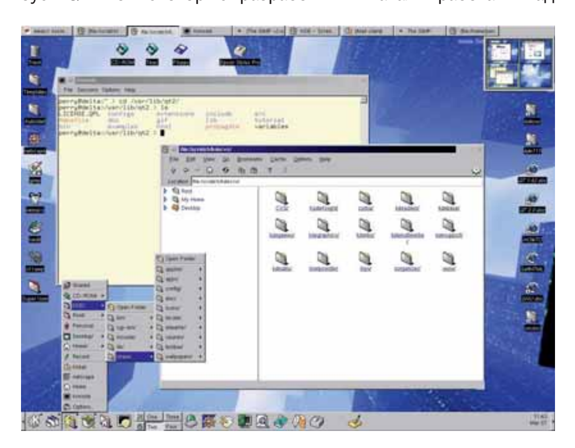
То, что каждый может присоединиться к проекту, – это реальная сила Linux и Open Source в целом. «Я скачал ядро 2.3.47 только для того, чтобы обнаружить, что Алан [Кокс] ушел и отметил мой NIC как устаревший! Поскольку я с некоторого времени интересовался разработкой ядра, я решил засучить рукава и исправить это». Вот так Эндрю Мортон [Andrew Morton] подключился к разработке Linux – сейчас он один из ведущих хакеров ядра и недавно начал работать в Google. Его опыт – типичный пример того, как многие открывали для себя Open Source. Ладислав Боднар [Ladislav Bodnar], создатель DistroWatch, рассказал нам, как он подключился к работе над клиентом электронной почты *KMail*: «Я хотел изменить работу некой кнопки. Скачал исходники, изменил соответствующий код, затем пересобрал и установил его. И заработало! Вот тогда я по-настоящему поверил в гибкость открытого ПО».

Дуэль рабочих столов

В этот период на передний план вышли многие из хорошо известных ныне имен. Появился Mandrake, основанный в 1998 году как ответвление от Red Hat (подробнее о Mandrake – через страницу). SUSE, выпустившую свой корпоративный дистрибутив v4.2 в 1996 г., многие воспринимали как европейский Red Hat. Но это было не только время Linux в серверном секторе: настольная версия становилась все более жизнеспособной. Хотя X Window был доступен в Linux с 1992 г. и входил в состав первых дистрибутивов, требовалось более функциональное окружение рабочего стола, способно переманить пользователей Windows.

Как, похоже, часто случалось в истории Linux, KDE начал жизнь с сообщения в Usenet. В статье, опубликованной в октябре 1996 г. на **comp.os.linux** под заголовком «New Project: Kool Desktop Environment (KDE)», немецкий студент Маттиас Эттрих [Matthias Ettrich] высказал свои замечания по поводу существовавших тогда рабочих столов, особенно указав на несогласованность и высокую цену. «По моему скромному мнению, графические интерфейсы должны предлагать завершенное графическое окружение», писал он. «Это должно дать возможность пользователям (!) выполнять их повседневные задачи: запуск приложений, чтение электронной почты, настройку своего рабочего стола, редактирование файлов, удаление файлов, просмотр изображений, и т.д. Все части должны соответствовать друг другу и работать вместе». Полный текст сообщения занял бы четыре страницы *Linux Format*, но вы можете прочитать его полностью на **http://snipurl.com/x7x8**.

Эттрих решил использовать для построения KDE инструментарий *Qt*. Он давал несколько преимуществ программистам, желающим писать большие, стройные приложения за довольно короткое время. Но у *Qt* был один недостаток – закрытость. Это обеспокоило тех, кто желал оставаться верным принципам GNU и свободного ПО. Торвальдс, всегда прагматичный, был вполне доволен KDE и тем, что он использует *Qt*. Но некоторые разработчики начали работать над



› **Собственный рабочий стол KDE 1.1 Маттиаса Эттриха.**

Что сталось с...

Indrema?
Анонс в 2000 году игровой консоли, основанной на открытом ПО, осуществил мечты многих фанатов Linux. Однако рынок игровых консолей строится на концепции продажи вашего оборудования по себестоимости или даже ниже и возмещении потерь за счет лицензий на игры. Если кто угодно может писать и выпускать ПО для вашей машины, лицензионные поступления не осуществляются. Этому достойному похвалы проекту было суждено так и остаться лишь фантомом.

Ключевые даты

Февраль 1997. Ричард Столлмен не одобряет KDE Маттиаса Эттриха (E), использующего *Qt* как основной инструментарий, и вынашивает планы по созданию более дружественного с точки зрения GPL окружения рабочего стола.



E

Август 1997. Мигель де Икаса (I) основал проект Gnome (GNU Network Object Modelling Environment) после неудачной попытки убедить Trolltech, разработчика *Qt*, принять двойное лицензирование.

I

Апрель 1998. Netscape, в отчаянной попытке выдержать бешеную атаку Internet Explorer, открывает код Netscape Navigator 5.

Июль 1998. Выпущены Debian 2.0 и KDE 1.0; база данных промышленного уровня Informix портирована на Linux.

Сентябрь 1998. Позорные «хеллоуинские» документы утекают из Microsoft, раскрывая методы, которыми компания



инструментарием *Harmony*, свободной и совместимой альтернативой *Qt*, который так никогда и не был закончен. Другие переключились на конкурирующий проект Gnome, основанный Мигелем де Икасой [Miguel de Icaza] и Федерико Мена [Federico Mena] в августе 1997 г.

По мере взросления проектов, Интернет объяло пламя войн «KDE против Gnome», подчеркивая раздробленность и пристрастность, характерные для Linux. Как и следовало ожидать, когда проект раскалывается по таким фундаментальные вопросам, как лицензирование и свобода ПО, накал страстей дошел до уровней, ранее замечаемых только в спорах *Vi* против *Emacs*, битвах Atari против Amiga да на футбольных матчах «Манчестера Юнайтед» с «Арсеналом». Ну, это, может быть, крайность – все же они не были столь яростными, как «*Vi* против *Emacs*». В 2000 году Гаэль Дюваль [Gael Duval] из Mandrake призвал к перемирию: «Почему мы уподобляемся традиционным производителям ПО? Все люди разные: у каждого свои потребности. Давайте объявим, что стандартом являются и KDE, и Gnome, и любая другая свободная высокоуровневая среда рабочего стола, достаточно хорошая, чтобы сделать Linux более подходящим для *всех* пользователей».

Проблема лицензирования отошла на второй план, поскольку *Qt* вышел под собственной Q Public License в 1998 году, с переводом Unix-версии на GPL в 2000-м. В наши дни обе организации участвуют в Freedesktop.org и наслаждаются значительной совместимостью. Топор войны был зарыт – но соперничество не умерло.

Спасение Netscape

Одно из наиболее важных событий этой эпохи произошло за сценой открытого ПО. В январе 1998 г. Netscape Communications Corporation выпустила исходный код своего титульного браузера, создав Mozilla Organisation, позже ставшую Mozilla Foundation. Это был весьма важный ход для Open Source, а следовательно, и для Linux. Самый факт спуска столь известного продукта на воду Open Source любимцем первоклассных дот-комов повысил понимание этого метода разработки, а финансовое обязательство, выданное Netscape вместе с этим релизом, предоставило финансирование также и разработчикам Open Source.

Похожее по важности событие произошло в октябре 2000 г., когда Sun открыла код своего *Star Office* под лицензией GPL и создала проект *OpenOffice.org*. При анонсе *Netscape* исполнительный директор Netscape Джим Барксдейл [Jim Barksdale] сказал: «Мы думаем, что это разительно изменит способ, которым люди фактически разрабатывают эти продукты, на многие последующие годы. Это станет историческим днем в данной цепи событий». Sun и Netscape, несомненно, продемонстрировали, что коммерческие компании могут создавать открытые проекты. *Firefox* и *OpenOffice.org* – две наиболее популярные открытые программы, на любой платформе.

Эти проекты дали возможность Open Source проникнуть в новые области, что доказывается и множеством людей, использующих их на Windows, и их поддержкой операционной системой Mac OS X. Наличие одного и того же ПО на Windows и Linux устраняет одно из препятствий для предприятий, переходящих на Linux, поскольку их пользователи уже знакомы с ключевыми программами. Вместе с растущим использованием программ, основанных на браузерах, это значительно снижает необходимость переобучения персонала. В некоторых случаях предприятия сообщают, что пользователи даже не замечали смены ОС. »

Интервью: Нат Фридмен

Нат Фридмен присутствовал рядом с Мигелем де Икасой при рождении проекта Gnome. Они основали Ximian для разработки рабочего стола Gnome, а в 2003 году компания была приобретена Novell. Фридмен продолжает работать над Gnome и сейчас.



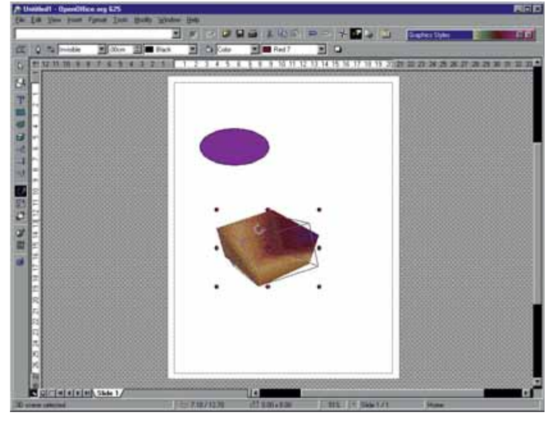
LXF: Когда Вы впервые столкнулись с Linux?

НФ: Впервые я запустил Linux в 1993 году на 386 машине, которую купили мне родители. Его показал мне мой друг Эдвард Лопер [Edward Loper]. Мы съездили на велосипедах в Университет штата Вирджиния, скачали образы SLS, записали их на 3,5-дюймовые дискеты и отвезли их домой в рюкзаках. Тогда нужно было использовать *Norton Disk Editor*, чтобы сделать ваш Linux-раздел загружаемым. В Linux была поддержка удаленного TTY, и я собрал кабели RS/232 из телефонного провода RJ-11 и поставил терминал в комнате своей сестры, так что она могла получать электронную почту и использовать Gopher и IRC. У меня также был калькулятор HP48, на котором работал эмулятор VT100. Мы поставили его в ванной, и прокинули туда кабель, и можно было болтать по IRC из туалета.

Linux поставлялся с компилятором C++, а я как раз изучал объектно-ориентированное программирование. И в то время существовал почтенный *xspringies* Дуга ДеКарло [Doug DeCarlo], который, похоже, уже почти забыт. Я нашел пару людей, используя трюк с привязкой к телефонным номерам статических IP, и у некоторых из них был Linux; я пользовался *ytalk* для контактов с ними на их компьютерах. Одним из них оказался Джеф Апхофф [Jeff Uphoff], который управлял списком рассылки по безопасности Linux, и мы скоро подружились. Думаю, мне было тогда лет 15 или 16. Это было невероятно забавное время.

LXF: Как появился Ximian?

НФ: Я потратил четыре года, по общему мнению – работая над дипломом, но в основном изучая Linux и Open Source и знакомясь со множеством людей из того мира. В летнее время и в различные перерывы я прошел несколько коротких стажировок в Microsoft, SGI, Media Lab и Red Hat. Я знал, что, получив высшее образование, захочу основать свою фирму. Каждый раз, когда я мог, я заходил на Linux-конференцию. Я встретил этого невероятно энергичного мексиканца по имени Мигель де Икаса в IRC, навещал его пару раз в Мехико, и мы подружились. Когда я закончил учебу, для нас было совершенно естественным основать фирму вместе.



› **Портрет раннего OpenOffice.org Draw.**

Знаете ли вы?

Имя Linux произошло от Linus + Minix, но Линус позже отверг имя Lignus, предложенное как комбинацию Linus и GNU. Это имя могло бы уберечь нас от всех этих разговоров «Это GNU/Linux, а не просто Linux» да «Как произносится Linux?».



намеревается бороться с угрозой Open Source.

Декабрь 1999. Просто чтобы показать, что Linux никак не защищен от безумия дот-комов, VA Linux побивает все рекорды отрасли своей первой продажей акций.

Апрель 2000. Открыт Minix, прародитель Linux. Также в апреле был основан Linux Format (вслед за тестовым Linux Answers) с Ником Вейчем у руля. Заголовок его новостей гласил: «Corel становится ‘агрессивным’ с Linux».

Сентябрь 2000. Вслед за выпуском Qt для Linux под GPL Столлмен и Free Software Foundation «отпустили грехи» тем, кто осквернил GPL своим вкладом в KDE.

Январь 2001. На замену полному ошибок релизу 2.2 выпущено ядро 2.4.



2001–2004 Тяжелые времена

После бума настал спад. Как Linux выжил?



Если какая-нибудь компания доказывает изменчивость фортуны, так это MandrakeSoft (ныне Mandriva). Ее вдохновитель и соучредитель Гаэль Дюваль начал работать с Linux, по его словам, «переводя некоторые HOWTO и Linux-документацию». Через год он выпустил законченный дистрибутив. Mandrake Linux 5.1 был выпущен в июле 1998 года как «KDE-фицированная» версия Red Hat, сохранив тот же номер версии, что и релиз Red Hat, на котором он был основан.

Mandrake вскоре приобрел свою индивидуальность, во многом благодаря всеобъемлющему набору удобных инструментов конфигурации и одному из самых простых инсталляторов. Он получил репутацию дистрибутива, дружелюбного к новичкам, хотя пригодного и для опытных пользователей. Mandrake был также славен использованием последних, даже передовых свободных программ, хотя это подчас подрывало стабильность.

А потом бум дот-комов закончился, и руководство Mandrake приняло ряд неудачных решений. Чтобы предупредить спад в бизнесе, они вложили средства в другие области, наняв при этом дополнительный персонал (и увеличив расходы). Фактически, основной бизнес продолжал приносить прибыль, зато вложения опустошали финансовые резервы. MandrakeSoft начала процедуру защиты от банкротства (*redressement judiciaire* – сохранение деятельности неплатежеспособного предприятия под судебным надзором) во французских судах в начале 2003 г. Но спустя 14 месяцев судебной защиты, MandrakeSoft смогла начать получать прибыль и создала план выплат долгов своим кредиторам, утвержденный судом. Это был урок ценой в миллионы евро: «не сломалось – не чини».

> «Это был урок по теме "Не сломалось – не чини" ценой в миллионы евро.»

Платное участие

Одним из факторов возрождения Mandrake был Mandrake Club, сетевой сервис для сообщества энтузиастов, действующий на принципах подписки. Это был разумный способ наладить прямую связь (и потоки прибыли) между компанией и пользователями. Дюваль недавно сказал нам: «Я думаю, обратная связь от сообщества была ключевым фактором, который помог нам не остановиться вообще и сказать: «Что ж, мы сдаемся». Вскоре после этого компания выпустила Mandrake Linux 10.0, вероятно, лучший дистрибутив того времени.

Дистрибутивы Linux и их изготовители вереницей сменяли друг друга за десяток лет после рождения Linux, многие появлялись без особых фанфар и еще незаметнее умирали. Одной из них могла бы стать Caldera Systems, основанная в 1994 г. Ее основным продуктом был Caldera Linux, дистрибутив, нацеленный на бизнес-пользователей, и она также между делом приобрела ряд продуктов Unix. К 2001 г. финансовые аналитики предупреждали, что компания еле держится на плаву.

В 2002 г. основатель Caldera и исполнительный директор Рэнсом Лав [Ransom Love] оставил компанию, и его заменил Дарл МакБрайд [Darl McBride]. После перемены имени компании на SCO, МакБрайд показал, что ее уход с рынка Linux будет далеко не тихим, из-за тяжбы с IBM в 2003 г. с суммой иска в миллиард долларов за использование «ворованного» кода в частях ядра Linux (IBM участвовала в разработке ядра 2.6). Хотя МакБрайд не упускал возможности заявить, что Linux содержит «код ядра Linux строка в строку совпадает с нашим кодом UnixWare», доказательств было представлено очень мало. В августе 2003 г. Торвальдс сказал, что компания, должно быть, «обкурилась крэком». В феврале 2004-го Торвальдс рассказал Business Week Online немного более подробно, что это абсолютно безосновательные заявления. «Есть буквально несколько уровней, где SCO заблуждается», говорил он. SCO, в конце концов, предложила пользователям возможность избежать исков при использовании «грязных» версий Linux, купив лицензию за 699 долларов на машину. Рассмотрение дела в суде назначено на февраль 2007 г.

SCO: послевкусье

Необходимость защищаться от инсинуаций затронула и тех, кто участвовал в разработке Linux, и сам процесс разработки. Никто не хотел быть обвиненным в воровстве или доказывать свою невиновность, когда дел и так по горло. Когда мы спросили Памелу Джоунс [Pamela Jones], редактора Groklaw, как эта тяжба влияет на Linux, она сказала нам: «Это раздражает и создает стресс. Представляете, что чувствует Линус, когда его проект ни с того ни с сего обвиняют в воровстве? Нужно волноваться, внимательно просматривать код, записи участников, беседовать с юристами, давать показания и так далее. Хорошего мало». (*Читайте полное интервью с Памелой Джоунс во врезке справа*).

Пока Mandriva боролась за возрождение, SUSE была обласкана Novell, купившей ее в январе 2004 г. «Никакой другой корпоративный поставщик Linux не имеет опыта



Что сталось с...

United Linux?
Образованный в 2002 году, United Linux был попыткой создать стандартный основной дистрибутив, чтобы избежать раздробленности, затронувшей Unix и грозившей Linux. Это был консорциум SUSE (позже поглощенной Novell), Turbolinux, Conectiva (впоследствии слившейся с Mandrake) и SCO Group. Работа замедлилась после подачи SCO ее иска против IBM, и в начале 2004 года Novell объявила, что «нет больше смысла» в группе с тех пор, как SCO заняла столь агрессивную позицию.



Ключевые даты

Март 2002. Дэниел Роббинс [Daniel Robbins] выпускает Gentoo 1.0, Linux-дистрибутив, в котором весь стек ПО собирается из исходных текстов, чтобы с гарантией все оптимизировать для целевой системы.

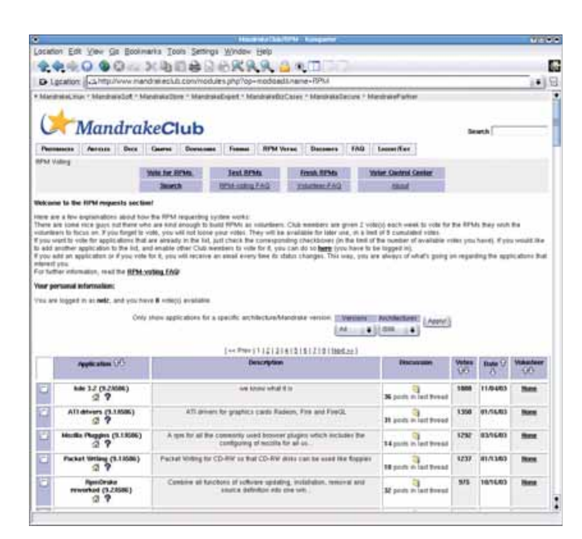


Май 2002. Объявлено о выходе *OpenOffice.org 1.0*, основанного на *Star Office* от компании Sun.

Март 2003. SCO – ранее Caldera Systems – объявляет о своем намерении возбудить дело против IBM о незаконном присвоении миллионов строк кода оригинальной системы Unix System V и перенесенных в Linux. Под руководством Дарла МакБрайда (D) SCO продолжит разборки с Autozone и Daimler Chrysler, Red Hat подаст в суд на SCO, SCO подаст в суд на Novell, Novell подаст в суд на SCO...



Июль 2003. Red Hat объявила, что уходит от продаж «коробочных» версий Red Hat для настольных систем



› **Возрождаясь, Mandrake основала Mandrake Club, предоставив пользователям возможность влиять на развитие дистрибутива..**

разработки ОС и возможности оказывать техническую поддержку по всему миру, которые может предоставить Novell», предсказывал Джек Мессман [Jack Messman], директор Novell, и как жест доброй воли, вся ИТ-система Novell была переведена на SUSE. «Не было никакого сопротивления ни внутри, ни со стороны клиентов», рассказывает Тони Данн [Tony Dunn], директор Novell по Linux в Великобритании, Среднем Востоке и Африке. «Внутренне, слияние двух компаний с различными культурами всегда требует осторожного менеджмента. Но в целом взгляды сотрудников из Novell и SUSE были очень позитивны. У некоторых клиентов возникали вопросы по поводу коммерческих аспектов открытой модели разработки; однако, получив объяснения, они понимали реальные выгоды и потенциал».

В Linux-сообществе были страхи, что приобретение ведущей корпорацией испортит SUSE; что она станет менее открытой. Фактически, произошло обратное: до этого приобретения *Linux Format* не мог поместить SUSE на свой диск, пока не получит особое разрешение, потому что и другие инструменты SUSE не были свободны для распространения. Novell пошла на выпуск их под GPL. Недорогой дистрибутив SUSE Personal был отброшен, но на его место (в 2005 году) пришел OpenSUSE, содержащий только свободное ПО.

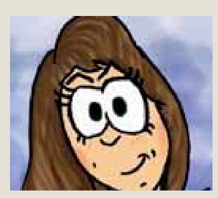
Вызов большим мальчикам

С дистрибутивами вроде Debian и Slackware, взывающими к пуристам и энтузиастам Linux и Open Source, и Red Hat и SUSE для коммерческих пользователей, оставалось захватить еще одну рыночную нишу. Это произошло в конце 2001 г. с выходом Lindows. По названию было ясно, как оно пытается позиционироваться, а если было недостаточно ясно, то у основателя компании был удачный рупор ее интересов в лице обладающего притягательной силой Майкла Робертсона [Michael Robertson], набившего карман продажей MP3.com компании Vivendi Universal за 372 млн. долларов. Целью Робертсона было производить дистрибутив Linux, способный делать все, что делает Windows, за счет умения запускать основные приложения Windows наряду с ПО для Linux (хотя юристы из Редмонда и вынудили его изменить название проекта на Linspire). Особой популярности среди Linux-пуристов он не получил, но для них он и не предназначался.

Робертсон крепко положил глаз на настольные системы – чтобы ОС Linux появилась на компьютерах обычных пользователей. Это стало следующей целью энтузиастов Linux; и кое-кто начал предсказывать, что «следующий год станет годом Linux на рабочем столе». В то время нельзя было найти ноябрьский или декабрьский Linux-журнал, где бы отсутствовал такой прогноз. Сейчас эти предсказания менее распространены, потому что либо людям уже набила оскомину эта фантазия, либо они осознали реальную ситуацию. Linux растет скорее эволюционно, нежели революционно. С каждым годом он хорошеет, каждый год привлекает больше пользователей, каждый год о нем узнают больше. Хотя, принимая во внимание популярность *Firefox* в последнее время, возможно, будет правдой сказать, что 2006 г. – это год настольных Linux-систем... »

Интервью: Памела Джоунс

Начатый в мае 2003 г., блог Памелы Джоунс (Pamela Jones), **Groklaw.net**, стал колючкой в боку SCO, с тех пор как она принялась обсуждать его судебные процессы. Получив неполное юридическое образование, Джоунс теперь журналист и ведущий защитник идеалов Open Source.



LXF: До процесса SCO у вас был большой опыт работы с Linux?

ПДж: Да. Поэтому я и заинтересовалась этим судебным делом. Мне нравятся компьютеры, и я постоянно использую несколько ОС. Так что когда SCO появилась на сцене, мне это определенно бросилось в глаза, потому что угрожало тому, о чем я заботилась.

LXF: Принес ли этот суд пользу Linux, способствуя пересмотру процедур, большей открытости процессов, и так далее?

ПДж: Процесс был несколько скорректирован, чтобы даже посторонним стала очевидна возможность отследить все изменения в коде, но правда в том, что это можно было сделать всегда. Ядро открыто и доступно любому 24 часа в сутки 7 дней недели, и вы можете отследить для себя все, произведя «обратный отсчет». Зато, я думаю, это помогло каждому осознать важность GPL и значимость работы с юристами до того, как произойдет неприятность, чтобы избежать ненужных проблем. Я считаю, что сотрудничество программистов и юристов – очень позитивное достижение. Теперь каждый знает, что бывают люди вроде тех, что в SCO, и нужно быть готовыми ко встрече с ними.

LXF: Что произойдет с Groklaw, если или когда этот суд наконец-то завершится?

ПДж: О, суд обязательно завершится, он только кажется бесконечным. Что касается Groklaw, то теперь это серьезное сообщество, и занимается множеством других дел. Я сфокусировалась на SCO, потому что это касалось всех и иллюстрировало ход судебного дела. Для этого нужен был затяжной процесс, и это было моей целью.

Так что мы, видимо, продолжим работу, освещая в новостях другие разбирательства. Хотя в конечном итоге это зависит от моих читателей. Когда я запустила Groklaw, предполагалось заниматься множеством судебных процессов, но однажды SCO стал доминирующим, поскольку читателей это очень волновало. Мы уже охватываем множество других тем, так что я ожидаю, что мы будем работать и дальше.

» Дарл МакБрайд был недоступен для интервью из-за приближающегося судебного заседания по делу SCO против IBM.

Знаете ли вы?

30 июня 2001 г. Дэвид Уилер [David A. Wheeler] опубликовал результаты исследования, показавшего, что разработка типичного Linux-дистрибутива обычными средствами обойдется в примерно в 1 млрд. долларов. Он выяснил, что Red Hat 7.1 содержит примерно 30 млн строк кода, отражающих более чем 8000 человеко-лет разработки.

и вместо этого запускает свободный, ориентированный на сообщество проект Fedora, как тестовый полигон для их корпоративных продуктов.

Август 2003. Novell покупает Ximian и ровно через три месяца заявляет о своих планах выйти на рынок дистрибутивов, приобретя SUSE.

Июль 2004. Microsoft улаживает свой спор с Lindows, заплатив компании 24 млн/ долларов и согласившись на использование некоторых медиабиблиотек Windows. Lindows становится Linspire.



Октябрь 2004. Выпущен Ubuntu, детище космического туриста Марка Шаттлворта [Mark Shuttleworth]. Продукт является ответвлением от Debian, и разработчики обещают выдерживать шестимесячный цикл разработки, как у Gnome.

Ноябрь 2004. Выпущен *Firefox 1.0*. Браузер основан на *Mozilla*, выросшей из пепла открытого Netscape *Navigator 5*.

Октябрь 2004. IBM использует свое эфирное время в трансляции Суперкубка для рекламы своего выбора Linux. Головокружительный ролик показывает Мохаммеда Али и маленького мальчика в белой комнате, который «изучает то, что изучаем мы».

2005–2006 Возврат к основам

Сообщество одумывается.



Дистрибутивы типа Red Hat и Debian производятся большими командами разработчиков, но бывает, что программист-одиночка делает дистрибутив (хотя бы и производный от другого), оказывающий значительное влияние. Это, несомненно, справедливо в отношении Slackware Патрика Фолькердинга. Slackware – не только старейший из выживших дистрибутивов, но также отец некоторых основных дистрибутивов и дедушка многих других.

Другой дистрибутив одного автора, впечатляющий по многим причинам – более современный Knoppix, написанный немецким программистом, ИТ-консультантом и пианистом Клаусом Кноппером [Klaus Knopper] (некоторые из его музыкальных сочинений размещены на www.knopper.net/music). Если вы не слышали о Knoppix, это LiveCD-дистрибутив: он загружается и работает непосредственно с CD, или DVD для более поздних версий. Его вообще не нужно устанавливать на ваш жесткий диск, хотя он способен использовать существующий раздел как домашний каталог для хранения документов и настроек.

Об определении этого первого LiveCD-дистрибутива можно спорить, в зависимости от того, что вы называете дистрибутивом. Считаете ли вы таковым CD-версию системы восстановления, предоставляющую только интерфейс командной строки? А как насчет оценочного диска SUSE? Так или иначе, Knoppix признан первым «правильным» LiveCD: он загружает графический рабочий стол с набором приложений. Но действительно работоспособным Knoppix сделала его система распознавания оборудования. На CD нельзя было сохранить основные установки, так что системе приходилось определять и настраивать ваше оборудование – экран, разделы жесткого диска, сеть, звук и многое другое – при каждой загрузке.

> «С появлением новых Live-дистрибутивов, команды старых испытали стресс.»

Карманный Linux-компьютер

Как и многие другие проекты, Knoppix был создан из любопытства. «Я хотел разобраться, как работают загрузочные CD, а когда базовая система заработала, я добавил кое-что лично для себя, например, распознавание оборудования и автоматический запуск предварительно настроенного рабочего стола». Кноппер говорил Ладиславу Боднару в 2002 году: «Когда вы преподаете информатику, ПК студентов не всегда настроены как надо. Так что для меня наличие загрузочного CD с полной установкой многое упростило». Проект никогда не предназначался для публичного выпуска, но тем не менее приобрел популярность как переносной компьютер, система восстановления и тестер совместимости оборудования, и к середине десятилетия стал столь же известен среди домашних пользователей Linux, как Mandriva, Fedora и SUSE.

При простом менеджере пакетов Debian и множестве приложений, Knoppix легко было модифицировать и подгонять под свои нужды (см. LXF74/75). Это стало развлечением для многих, и в 2005 г. новые дистрибутивы на базе Knoppix появлялись чуть ли не еженедельно. Вскоре многие из них, да и сам Knoppix, начали дополняться инсталляторами для установки на жесткий диск, стирая грань между LiveCD и традиционными дистрибутивами. Старые проекты ощутили стресс: если можно загрузить LiveCD, типа SimplyMepis или PCLinuxOS, увидеть, что все работает, и затем просто щелкнуть по иконке, чтобы установить его, захочется ли пробовать что-то еще? В результате другие дистрибутивы сейчас заменяют свои установочные диски на LiveCD (Mandriva и Ubuntu), или предлагают вариант LiveCD для вас, чтобы его можно было сначала попробовать.

Еще одна ударное достижение последних двух лет – готовность изготовителей встраивать Linux в самые разные устройства, например, в хорошо принятый планшет Nokia 770. Trolltech, компания, стоящая за *Qt*, сыграла огромную роль со своей платформой Qtopia. Бенуа Шиллингс [Benoit Schillings], ее технический директор, говорит, что Trolltech «увидела заметный сдвиг в принятии Linux производителями устройств» в прошедшие несколько лет: «Мы тесно сотрудничали с другими первопроходцами Linux, назову хотя бы Motorola, ZTE, Datang и Kangaroo TV, сделавшими Linux жизнеспособной платформой в области, где прежде выбор был только среди Microsoft, Symbian и проприетарным ПО».

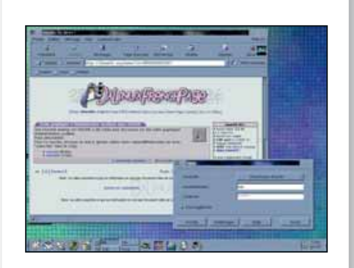
Дистрибутив или образ жизни?

Хорошие дистрибутивы Linux зачастую управляются сообществом; они появляются, люди пробуют их, рассказывают своим друзьям, затем те пробуют их. И если дистрибутив хорош, молва о нем распространяется быстро. Именно так было с Knoppix, и так случилось и с Ubuntu. Трудно поверить, что Ubuntu менее двух лет от роду (первым релизом был 4.10, номер версии отражает год и месяц выпуска). Сегодня Ubuntu находится на верхней строчке рейтинга DictroWatch.com, на



Что сталось с...

DemoLinux?
DemoLinux был LiveCD-дистрибутивом, на три года опередившим Knoppix. И он с полным основанием мог бы претендовать на звание первого в этом роде: загрузка с LiveCD полноценного графического рабочего стола. В версии 2.0 даже был *Star Office* и выбор Gnome или KDE. Проект был активен в течение года, пройдя за это время путь от 1.0 до 3.01. С тех пор выпусков DemoLinux больше не было, но он все еще доступен на www.demolinux.org.



Ключевые даты

Январь 2005. Профессор Массачусетского технологического института Николас Негропонте объявил на Международном экономическом форуме в Давосе (Швейцария) об амбициозном проекте создать ноутбуки по цене 100 долларов для детей в развивающихся странах. Ученые мужи разумно предположили, что только Linux предоставляет преимущества стоимости и гибкости, способные сделать этот проект реальностью.

Февраль 2005. Заместитель командующего по ядру Эндрю Мортон (**A**) объявляет, что будущая версия текущего ядра 2.6 будет интегрирована с ПО виртуализации Xen.

Апрель 2005. Mandrake (у которого были проблемы из-за товарных знаков Hearst Corporation) объединяется с бразильским поставщиком дистрибутива Conectiva и становится Mandriva.

A




Август 2005. Novell запускает проект OpenSUSE, который, подобно Fedora и Red Hat, будет работать как учебный полигон для корпоративных продуктов компании. Novell надеется влиться в сообщество разработчиков Linux, чтобы улуч-



› **Дистрибутив Ubuntu стал суперпопулярным всего за 18 месяцев, и он поощряет ответвления вроде Edubuntu.**

момент написания статьи почти на 40% опередив дистрибутив OpenSUSE, занимающий второе место. За свою короткую жизнь Ubuntu завоевал массу премий и наград, и его популярность продолжает расти.

Марк Шаттлворт, основатель Ubuntu, думается, вложил в проект немало денег, но это не единственная и даже не основная причина его стремительного взлета. Другие применяли подобный подход гораздо менее успешно. Причина успеха Ubuntu, по мнению авторов данной статьи, в понимании того, что нужно людям – и инициативные пользователи ценились гораздо больше, чем финансовые вливания. К 2005 г. популярные дистрибутивы раздулись с обычных в 2001 г. двух дисков до пяти дисков или DVD. Единственный CD-диск Ubuntu содержал все необходимое для начала, имел простой процесс установки и единственный рабочий стол (Gnome), привлекая тех, кто не желал устанавливать заодно с дистрибутивом всякие экзотические примочки.

Подобно Knoppix, Ubuntu основан на нестабильной ветви Debian. Он исправил основные неудобства Debian – недружественную инсталляцию, растянутый цикл обновлений, устаревшие пакеты – и сохранил его хорошие качества.

Но феномен Ubuntu – это больше, чем версии ПО и инсталляторы. Опросите группу пользователей Ubuntu (или любого из его клонов, типа Kubuntu или Edubuntu), что им нравится в нем больше всего, и в большинстве ответов прозвучит слово «сообщество». Ubuntu – дистрибутив, реагирующий на потребности своих пользователей, да и направляется ими же, с чем трудно справиться коммерческому дистрибутиву. Когда мы спросили Марка Шаттлворта, на каких пользователей ориентирован Ubuntu, он сказал: «на разработчиков и людей, которые о компьютерах мало знают и знать не хотят, а хотят иметь средство для своей работы – где они без усилий найдут то, что им нужно». Звучит как противоречие, но на самом деле его здесь нет: именно смесь разработчиков и обычных пользователей обогащает сообщество.

Новый менеджер сообщества Ubuntu в Canonical Ltd. (официальный спонсор Ubuntu), ветеран-пропагандист Джоно Бэкон [Jono Bacon] должен запрягать и подхлестывать эту смесь пользователей. И мы думаем, он с этой работой справляется. Бэкон начал эксперименты с Linux в 1998 г. – затем, как он говорит, «однажды ночью – эврика! – я осознал потенциал исследований при участии в сообществе. От возбуждения я не смог уснуть и потратил всю ночь, записывая в блокнот, что я мог бы сделать и как улучшить свободное ПО».

В последнее время наблюдается ободряющий возврат к основам. Популярность однодисковых дистрибутивов, успех проектов-конструкторов «сделай сам» типа Gentoo и Linux From Scratch, теплый прием OpenSUSE и Fedora Core – все это демонстрирует саморегулируемость природы сообщества Linux. Если нас не устраивает направление движения, мы (или те из нас, у кого есть необходимые умения) можем отступить и перейти на другой образ действий. Таким образом, Open Source – это несколько больше, чем способность изменять работу программы: может круто поменяться и направление операционной системы, если этого захочет достаточное число пользователей, и неважно, какие решения приняты в залах заседаний.

Интервью: Джим Землин

Джим Землин [Jim Zemlin] стал исполнительным директором Free Standards Group в мае 2004 г., поработав как вице-президент по маркетингу в Covalent. Сформированная в 1998 г. FSG верит, что открытое ПО для дальнейшего развития следует стандартизировать. Среди ее проектов – Linux Standards Base.



LXF: Как, по-вашему, изменилось ли отношение людей к Linux за эти годы?

ДжЗ: Я участвую в движении Open Source с 1999 года. За это время рынок вырос из «новой» методологии разработки и лицензирования, применимой немногими, до движения, наиболее влиятельного на рынке, бросившего вызов всем предрассудкам, которые у нас были в отношении разработки и продажи ПО. Linux заменяет любой вариант Unix как основная угроза Microsoft.

LXF: Вашей целью является еще большая стандартизация, всегда критикуемая изнутри сообщества?

ДжЗ: На самом деле, нет. Разработчики Open Source, возможно, лучше чем кто-либо понимают, что стандартизация способствует инновациям. Без HTTP у нас не было бы Apache. Люди не хотят заново изобретать велосипед; они хотят что-то изменять и строить что-то новое и полезное. Все пришедшие к соглашению по базовому набору функций будут способствовать этой инновации.

Наша самая большая проблема, вероятно, та же, что и у любых других попыток стандартизации: поиск «золотой середины» между потребностью в стандартах и потребностью поставщиков отличаться друг от друга. Но в мире Open Source это напряжение, вероятно, более очевидно из-за необычайной скорости развития. Я бы сказал, что самый большой компромисс, достигнутый сообществом Open Source – это понимание важности стандартов, обратной совместимости И прочих вещей, который не слишком занятны, но жизненно важны для коммерческого применения.

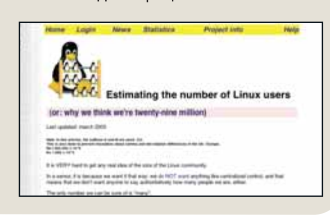
LXF: Какие основные проблемы ждут Linux в следующие два – пять лет?

ДжЗ: Думаю, важно, чтобы поставщики Linux продолжали поддерживать стандарты, типа LSB, чтобы разработчики приложений могли охватывать большие возможности Linux. Также нужна индустрия Linux, предоставляющая разработчикам больше инструментов и поддержки. Недавно был призыв к действию на eWeek.com, гласивший, что Linux должен предложить аналог Microsoft Developer Network. Это большая проблема для Linux, но я уверен, что она будет решена.

Также очевидно, что настольные системы Linux имеют немало насущных проблем, требующих решения: например, заставить печать «просто работать» для обычного пользователя. Я, опять-таки, уверен, что рынок обратит на это внимание.

Знаете ли вы?

На самом деле, никто не знает, сколько в мире пользователей настольных систем Linux. Windows и OS X могут сосчитать продажи лицензионных копий, но пользователи Linux могут получить дистрибутив через анонимные закачки, с журнальных дисков или от друзей. По различным оценкам, доля Linux на рынке ОС – от 1 до 5 процентов.



»

шить свои собственные продукты и «распространять весть о Linux».

Январь 2006. Ричард Столлмен и его команда юристов – прежде всего, Эбен Моглен (Э) – начали 12-месячные дебаты по обновлению GNU General Public License, чтобы она учитывала управление цифровыми правами (digital rights management) и патентами на ПО. Линус Торвальдс отказался переводить код ядра на новую лицензию.



Март 2006. Основатель Mandrake Гаэль Дюваль оставляет Mandriva упражняться в снижении затрат и основывает новый проект настольного дистрибутива под названием Ulteo.



Май 2006. Google объявляет о выпуске своего первого настольного приложения для Linux, *Picasa for Linux*. В следующем месяце удивительная *Google Earth* выпущена как полностью «родное» приложение.



Июнь 2006. Появился Ubuntu 6.06 LTS. Этот дистрибутив – первый релиз Ubuntu, объявленный пригодным для бизнес-применения.

Взгляд в будущее 15

Чего ожидать в следующие несколько лет?

Мы хотели бы завершить статью обсуждением, какие достижения и препятствия могут возникнуть у Linux в грядущие годы – но разве это предскажешь? Или мир Linux слишком извилист для точных прогнозов? Кто, например, мог представить 15 лет назад, что один довольно талантливый хакер, который возился с папашиным Vic-20, сможет создать нечто имеющее столь глубокий эффект? Кто мог предсказать, что миллиардер из Южной Африки, желающий улучшить образование африканских детей, почти вырвет рынок дистрибутивов из-под носа первопроходцев?

Как мы обсуждали на предыдущей странице, Linux направляется нуждами своего сообщества. В отличие от традиционного бизнеса, когда компания решает, какой продукт можно продавать для получения прибыли, и изо всех сил убеждает покупателей, что именно этот продукт им и нужен, Open Source реально дает пользователям то, что они хотят. Это среда, действительно ведомая потребителями, в противоположность среде, ведомой принципом «что, по нашему мнению, мы сможем всучить потребителям», потому-то развитие Linux столь захватывающе... и непредсказуемо.

Так что, не имея заслуживающего доверие хрустального шара, мы дадим вам нечто получше: предсказания людей, очень близких к Linux.

Что сталось с...

GP2X?

Этот отважный маленький «наладонник» поддерживается живой и энергичной группой. Появившийся в 2005 г., он обещает выполнять практически любое приложение, какое вам нравится, включая музыкальные и видео-плейеры, web-серверы и игры. На этот момент только *Vektar* выпускается на коммерческой основе, и хотя доступны различные свободные и открытые игры, есть надежда, что GP2X сможет принести достаточно дохода от лицензирования. В сентябре 2006 г. было объявлено, что его вторая игра, *Payback*, приближается к выпуску.



Джефф Во

До недавнего времени Джефф Во [Jeff Waugh] работал в Canonical. Он оставил свой пост в начале этого года, чтобы отдаться своей подлинной страсти: Gnome.

«Я думаю, вслед за выходом Vista в определенной степени подключатся и «партийные шляпы». Новая версия Windows выглядит мыльным пузырем, и, похоже, не сильно-то возбуждает у бизнес-пользователей желание обновляться сразу после релиза. Возникает вопрос: «Если обновление до Vista – большая морока, то почему бы не рассмотреть эту штуку по имени Linux, или другие варианты?» Я думаю, Apple останется значительной угрозой для FOSS, но «наши» компании готовятся к борьбе. Ubuntu 6.06 и SUSE Linux Enterprise Desktop 10 – превосходные продукты. Такие вещи, как улучшенное управление питанием и технологии отрисовки экрана, развиваются сейчас в Gnome очень быстро. Может, у Vista и есть преимущества в интерфейсе, но в терминах дерзкого, аппаратно ускоренного пользовательского опыта мир FOSS превосходит все предлагаемое в Vista».

Себастьян Кюглер

Себастьян Кюглер [Sebastian Kugler] – член команды по маркетингу KDE.

«За пять лет мы выйдем на основной рынок ПО, отхватив на нем примерно 10%. В этом месте поставщики оборудования и ПО начнут заботиться об активной поддержке Linux. Достоинства открытой модели разработки («стоя на плечах гигантов») теперь реально окупаются. Скорость разработки свободного ПО и качество продукта станут значительно лучше, чем проприетарного ПО.

Сейчас KDE готовит основу на следующую пару лет. Для рабочего стола KDE станет предельно просто разрабатывать полнофункциональные, полностью интегрированные приложения на множестве языков программирования. Новые креативные концепции станут результатом открытости свободного ПО для сочувствующих, не занятых непосредственно в разработке».

Гаэль Дюваль



Дюваль был одним из лидеров разработки Mandrake/Mandriva. Он ушел из компании в марте и готовит выход Ulteo, нового дистрибутива.

«В сфере серверов, я думаю, виртуализация с Xen и дополнительные средства безопасности (NSA SELinux, ядра RSBAC...) становятся горячее с каждым днем. На настольных системах самое последнее заметное достижение, по-моему, трехмерные средства для рабочего стола. Это забавно, и я не могу дождаться, пока они начнут работать со всеми видеокартами. Но появляются новые, менее заметные технологии, типа UnionFS и Fuse. Они очень многообещающи, потому что делают Linux еще более гибким и позволяют нам проектировать некоторые совершенно новые способы использования Linux (вроде тех, что можно найти в LiveCD). Так что я действительно думаю, что хотя многие дистрибутивы Linux весьма «консервативны» на настольных системах (помимо 3D-средств), мы теперь находимся на «пороге эпидемии», и

5 вещей, заслуживающих внимания



Firefox. Этот браузер становится все сильнее и сильнее, сбив в этом году рыночную долю *Internet Explorer* ниже 90%.

OpenSolaris. Ходят слухи, что Google начал тестирование последнего релиза пакета Sun OpenSolaris с прицелом на перенос своих 100 тысяч с хвостиком серверов с Linux.

Патенты на ПО. Грядет ли патентная буря? Большинство считает, что да, и соответствующим образом готовится. Для организации «No Software Patents» это означает жесткое лоббирование в Европарламенте, для продвижения перспектив свободного ПО. Для других, включая OSDL, IBM, Nokia и Red Hat, это означает накопление патентного арсенала.

Qtopia Greenphone. Продукт Trolltech даст совершенно новой когорте разработчиков мобильных приложений возможность принести фантастическое ПО в массы.

KDE 4. Это или очень храбрая, или очень глупая попытка пересмотреть практику рабочего стола. Технологии типа *Plasma* и *Phonon* и вправду обещают очень крутой графический интерфейс пользователя.

» Мы расскажем о KDE4 в LXF87/88

это обещает очень крутую и увлекательную разработку на следующие несколько лет».

Крис ДиБона



Крис ДиБона [Chirs DiBona] отвечает за Linux и Open Source в Google.

«Linux продолжит доминировать на рынке серверов и будет значительно вторгаться в мобильное пространство. Доля рынка настольных систем достигнет, возможно, уровня 3%, включая корпоративные и правительственные развертывания. Дальнейшая разработка Wine и Mono имеет потенциал для удержания миллионов людей от обновления до Vista, чреватого потерей обратной совместимости со своими приложениями. К тому же, я думаю, мы еще оглянемся на дебют Ubuntu 6.06 и SLED 10 как на поворотную точку в принятии Linux. Наконец, я думаю, что патенты на ПО резко атакуют Linux и другое ПО с [точки зрения] укоренившихся проприетарных интересов».

Блейк Росс



Блейк Росс [Blake Ross] был одним из ключевых разработчиков проекта браузера *Mozilla* и основал Firefox с Дэйвом Хайяттом [Dave Hyatt]. Сейчас он находится в годичном отпуске в Стенфорде.

«Работа с компьютерами достигла поворотной точки. В прошлом обсуждения фокусировались на внутреннем устройстве. Мы отвечали на вопросы, уходящие корнями в информатику: Как мы ускоряем эти вещи? Как мы повышаем эффективность многозадачности? Эти проблемы никогда не будут «решены» – всегда останется потребность сделать быстрее, мощнее, компактнее, но современные компьютеры могут без усилий обработать все, что в обозримом будущем понадобится среднему человеку.

Я думаю, что ответ лежит между ОС и Интернетом. Эти две чрезвычайно передовые платформы прекрасно дополняют друг друга, но фактически ничто их не связывает, что ставит людей перед безрадостным выбором. Хотите – создавайте контент с помощью мощных инструментов в свободном от рекламы окружении и схороните его среди хлама своей файловой системы, которая доступна в любое время, но не отовсюду, и только вам; а хотите – создавайте контент более слабыми инструментами и разместите его среди рекламы, и он будет доступен всем и везде, но только когда вы подключены к сети… Уже то, что концепция «загрузки» и «скачивания» существует на уровне пользователя, свидетельствует о проблеме, и я предсказываю, что она устареет через пять лет.

Сейчас самое время отложить внутреннее устройство и объединить эти миры с точки зрения потребителя. Кое-что я как раз сейчас делаю как часть нового проекта (www.blakeross.com/next).»

Нат Фридмен



Нат Фридмен работает в Novell и занимается разработкой как рабочего стола Gnome, так и Ximian.

«Open Source станет повсеместным. Понятно, что начинающая компания или компания, занимающаяся производством либо обслуживанием (типа TiVo или Google) будут пытаться сократить время выхода на рынок за счет привлечения открытых компонентов в свой продукт, чтобы не писать код заново и не платить за лицензии сторонним разработчикам. Эти предприятия не обязаны будут открывать свои продукты (да и не открывают), но они в любом случае получат преимущества от лицензии, не непременно участвуя в методологии разработки. Будет интересно наблюдать, как распределенная работа приживется в качестве модели.»

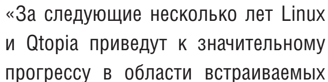
Йон 'Мэддог' Холл



Йон 'Мэддог' Холл [Jon 'Maddog' Hall] всегда с нами как пропагандист open Source. Мы спросили его, будет ли будущее принадлежать сетевым приложениям.

«Я сомневаюсь, что все приложения подходят для модели «ПО как сервис», так что воздействия, ожидаемого некоторыми, может и не быть. Есть конкретные приложения и конкретные потребители, которым будут нужны локальные приложения и локальные системы. Во-вторых, даже с ПО в роли сервиса, должно быть что-то, на чем этот «сервер» будет работать. Наконец, а кто-нибудь вообще спросил потребителей, хотят ли они ПО как сервис? Или это просто очередная идея, как наступить потребителю на горло? Чем плоха идея просто покупать то, что мне нужно, и использовать это тихо и мирно на моем собственном компьютере?»

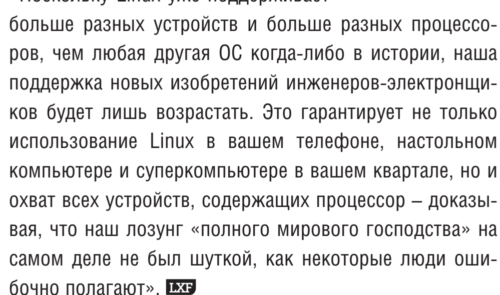
Бенуа Шиллингс



Бенуа Шиллингс [Benoit Schillings] был одним из сооснователей BeOS и помог запустить VoIP-сервис OpenWave. Он – технический директор Trolltech.

«За следующие несколько лет Linux и Qtopia приведут к значительному прогрессу в области встраиваемых систем. Производители устройств будут привлечены способностью этих технологий адаптироваться под навыки пользователя, выстраивать свою собственную лояльность к торговой марке, получать гибкость выбора из множества интегрированных опций сторонних производителей для дополнительной функциональности и развиваться на множестве моделей и типов устройств. Конечно, производители также получат преимущества от доступа к огромному сообществу наемных разработчиков и добровольцев».

Грег Кроа-Хартман



Грег Кроа-Хартман [Greg Kroah-Hartman] – разработчик ядра Linux, поддерживающий подсистемы PCI, USB, базу для драйверов и sysfs. Он работает в SUSE Labs в Novell.

«Поскольку Linux уже поддерживает больше разных устройств и больше разных процессоров, чем любая другая ОС когда-либо в истории, наша поддержка новых изобретений инженеров-электронщиков будет лишь возрастать. Это гарантирует не только использование Linux в вашем телефоне, настольном компьютере и суперкомпьютере в вашем квартале, но и охват всех устройств, содержащих процессор – доказывая, что наш лозунг «полного мирового господства» на самом деле не был шуткой, как некоторые люди ошибочно полагают». LXF

Знаете ли вы?



Основано более 800 проектов Linux-дистрибутивов. Многие из них являются ответвлениями существующих дистрибутивов, которые в свою очередь являются ответвлениями других. Некоторые специфичны для определенных областей, включая поддержку различных кодовых страниц или языков. Большая часть сейчас исчезла в результате того, что разработчик потерял к ним интерес или больше не существует потребности в его специализации.

Вливайтесь!

Что, по вашему мнению, является важнейшими моментами в истории Linux? Куда, как вы думаете, он идет? И каковы ваши самые сильные воспоминания об Open Source за последние 15 лет? Отправьте свое мнение и воспоминания на letters@linuxformat.ru.

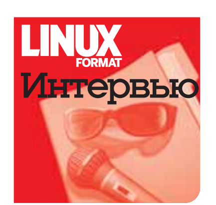


О PERL 6

«Он будет готов, когда будет готов, а если я назову вам дату, получится, что я соврал.»

Толкование Торка

Linux Format имеет эксклюзивную возможность объяснить, почему так задержался Perl 6: член совета директоров Perl Foundation **Нат Торкингтон** слишком занят… решением арифметических головоломок.



Когда Ной собирал всякой твари по паре, чтобы спасти от всемирного потопа, на его стороне был Бог. Когда Крысолов уводил из Гаммельна детей, у него имелись дудочка и волшебная мелодия. А вот Нат Торкингтон собирает компьютерных фанатов со всего мира на конференцию OSCon, имея только web-сайт, электронную почту и немного денег от Тима О'Рейли – а задача-то ничуть не проще. Когда **Грэм Моррисон** встретился с председателем программного комитета OSCon, он сперва решил поговорить с Торкингтоном о другой его колоссальной задаче: Perl.

Linux Format: **Хватает ли вам времени на программирование?**
Нат Торкингтон: Последнее время не хватало. Я был очень, очень занят работой [редактора] у O'Reilly, но в этом году стал уделять все больше и больше времени прошлым занятиям. Так что я освоил Ruby, и мне это нравится. Я работаю над головоломками конкурсов для программистов и снова балуюсь с кодом.

LXF: **В девяностых вы учились и преподавали в Университете Виктория в Веллингтоне, Новая Зеландия. Так и произошло ваше знакомство с Perl?**
НТ: Ну, типичная необходимость любого университета – виртуальный список для поиска имен. В то время единственный способ реализовать это был через web-сервер на Perl под названием Plexus. И мне пришлось выучить Perl, чтобы работать с ним. Я изучал Perl, начиная с *Programming Perl,* самой первой книжечки, с розовым верблюдом, и это было здорово. Я учил С по книге Кернигана и Ричи [*Kernighan & Ritchie, C Programming Language*], так это было гораздо проще, чем С. [Смеется.] Много, много проще, чем С.

LXF: **Вы считаете, что PHP украл часть славы Perl?**
НТ: Естественно. Этот язык занимался предоставлением легкого способа создания динамических web-страниц, и те, кто раньше обращались к Perl из-за динамических web-страниц, теперь обращаются к PHP. Но это нормально. Я не думаю, что эти языки до сих пор соперничают.

На вершине пищевой цепочки создатели этих языков, такие, как Гвидо [Guido van Rossum] из Python, Расмус [Rasmus Lerdorf] из PHP, Ларри [Larry Wall] из Perl, они все уважают друг друга, общаются, им нравится то, чем занимаются другие. Иногда конкуренция возникает среди пользователей, может возникнуть среди программистов, но даже это происходит все реже и реже.

LXF: **Вам не кажется, что это благодаря свойствам мира Open Source?**
НТ: Вы о конкуренции или о ее уменьшении?

»



› OSCon 2007 займется поисками докладов, начиная с ноября – не забудьте! http://conferences.oreillynet.com

» **LXF: Я об уважении и открытости. Например, вы изучаете Ruby.**
HT: Я думаю, что это характерно для мира технологии. Ребята из AT&T Bell Labs, которые изобрели Unix и C и еще многие технологии, сейчас уже воспринимаемые как нечто само собой разумеющееся, они точно так же открыты для умных людей и для умных систем, созданных IBM, или Линусом Торвальдсом, или кем-то еще. Если это – умная разработка, если она чего-то стоит, то ее уважают.

LXF: Долго ли еще ждать выхода Perl 6?
HT: Помнится, была такая шутка, что это произойдет под Рождество. Просто мы пока не знаем, под какое именно Рождество.

LXF: Думаю, вас об этом спрашивают каждый раз при разговоре о Perl.
HT: Точно. Могу сказать, что релиз будет готов тогда, когда он будет готов. А если я назову вам дату, то получится, что я соврал. Произойдет это раньше, или позже – в любом случае я не буду в выигрыше.

LXF: А вообще-то нужно, чтобы был релиз?
HT: А, значит, вы считаете, что это такой идеал вдохновения – даже если мы никогда не сделаем релиза Perl 6, все равно у нас будут продолжать работать над ним.

LXF: Но он вроде неплохо работает и такой как есть – книг по нему хватает.
HT: Ну, поиграть с Perl 6 можно уже сейчас. Можно побаловаться с системой *Pugs*, это интерпретатор Perl 6 на основе Haskell. Система интересная; Ларри пользуется ею как основной платформой для работы над остальной частью языка. Когда Одри [Audrey Tang] создала *Pugs*, Ларри мог сказать: «Все это уже сидело у меня в голове, давайте-ка напишем несколько тестов». А потом он начал осознавать: «Ого, в реальной жизни все сложнее, как бы это упростить?». А когда все это материализовалось и было проверено экспериментально, начал вырисовываться дизайн. Сейчас мы находимся в той стадии, когда некоторые портируют различные функции Perl 6 назад в Perl 5, и это по новой заставило сообщество разработки Perl 5 впрячься в работу.

LXF: Должен сказать, я ничего не знаю о Haskell. Почему выбрали его?
HT: Я ничего не знаю о том, почему был выбран Haskell, вам, наверное, надо поговорить с Одри, но мне известно, что для данной задачи он подошел отлично. Вчера вечером я разговаривал с Одри и Максом [Max Maischein, программист Perl], и она сказала – все так здорово, что она будет пользоваться этим и для любых проектов в будущем.

LXF: Выглядит заманчиво, мне нравится эта идея.
HT: Это похоже на Perl 6. Perl 5 был интересен, потому что перенес технологию из академической сферы в повседневную жизнь, сделал ее доступной для всех, и многие впервые столкнулись, например, с замыканиями в Perl. У Python ушло много времени на создание замкнутых выражений, которые были истинными замкнутыми выражениями. Программисты Perl могли работать с ними и изучать этот аспект программирования. Если вы программируете на Lisp, это ваш хлеб с маслом, вы выросли на всем этом. Но если вы – программист, начинающий с азов, то с неочевидными концепциями, не встроенными в язык, вы вряд ли встречались. Perl всегда много работал над обеспечением доступности концепций, стоящих как бы на шаг в стороне от языков массового применения, и Perl 6 продолжает ту же политику. Я думаю, что Haskell на данный момент как раз на шаг в стороне от Perl 6. В нем есть концепции, которых Ларри на данный момент побаивается.

LXF: Значит, не так уж важно, что Pugs использует Haskell?
HT: Нет, нисколько. Haskell оказался очень интересной и быстрой платформой разработки для концепций Perl 6. *Pugs* потому и был написан очень быстро, что писался на Haskell. Если бы его писали на C, мы бы все еще пытались добраться до стадии «*Hello World*».

LXF: Отражает ли возрастающая популярность объектно-ориентированного программирования желание людей программировать именно таким образом, или говорит о чем-то еще – об изменении направления для Perl?
HT: Perl включал объекты, начиная с выхода Perl 5 – думаю, это были 93, 94-й годы [Perl 5 вышел в 1994] – так что в Perl всегда была возможность создания объектов и вызова их методов.

LXF: Однако же это становится более формализованным.
HT: Вот именно, становится более формализованным. Я изучал объектно-ориентированное программирование [ООП] на Perl. В университете я занимался объектно-ориентированным программированием, но оно не привилось.

О HASKELL
«Тут есть концепции, которых Ларри на данный момент побаивается.»

LXF: Значит, вы не переходили с C на C++?
HT: Нет, наш выпуск последним работал на C, а потом перепрыгнули сразу на Java. На первом курсе мы работали на Pascal, а он еще более варварский.

LXF: А мы работали на Modula-2.
HT: Мы тоже работали на Modula-2, на втором курсе, чудаки, правда? Мы использовали *Metrowerks C Compiler*, мерзкую и жуткую программу.

Perl всегда поддерживал ООП. Один из уроков, усвоенных нами, заключался в том, что все должно быть проще. ООП – концепция высшего уровня, о таких сейчас много говорят. Ее нельзя эмулировать, ее нельзя разделить на более мелкие части, чтобы потом другие собирали объекты из этих мелких частей – а это вы на данном этапе и имеете в Perl, где метод является функцией, структура данных может ассоциироваться с классом, и все это превращается в объект. Сейчас очень хорошо видно, как это работает внутри. Perl 6 говорит: «Возьмите объекты и сделайте их главными, гражданами первого класса, чтобы вы могли программировать прямо в объектах, не беспокоясь о внутренних деталях». Хотя Perl – это Perl, внутренние детали там будут, если понадобится.

LXF: Вы бы стали использовать объекты для быстрого скриптинга, или же будете использовать Perl, как вы всегда делали?
HT: Объекты для быстрого скриптинга в Perl 5 использовать по-прежнему можно – сейчас многие функции проявляются через объекты. Например, самый быстрый и простой способ создать сетевое подключение на основе сокета TCP/IP, – это использовать класс IO::Socket. Одна строка дает вам объект, куда затем можно печатать, или откуда можно читать, как при работе с файлами. И это проще, чем делать так, как раньше делалось в Unix, старомодным способом: создать сокет, ассоциировать с ним протоколы, привязать его к порту, и… р-р-ррр.

LXF: Я хотел спросить у Элисон [Allison Randall]: а где место Ponie?

НТ: Ponie стал важным уроком по внутренним структурам Perl 5 и Perl 6, и в том, как заставить их работать вместе. Люди из Perl 5 постоянно что-то заимствовали оттуда, и люди из Perl 6 тоже, пока мы не подумали: «Какой смысл вести отдельные проекты? Надо работать над Perl 5 и Perl 6 параллельно». Так что сам Ponie больше не будет отдельным проектом, но вся проделанная работа войдет в Parrot или Perl 5. Ponie дал возможность заглянуть в будущее, увидеть разницу в мировоззрении Perl 5 и Perl 6, и понять, как они стыкуются.

LXF: Я много читал о том, что Perl 6 развивается благодаря сообществу пользователей. Кроме того, что это удлиняет процесс, как это влияет, например, на ратификацию стандартов? Важно ли то, что проект развивается благодаря идеям всего сообщества, а не одного-двух человек?

НТ: Трудно сказать. Трудно ответить, потому что другого способа разработки программ с открытым кодом нет.

LXF: Но для Perl это переход.

НТ: Большим отличием стало то, что в Perl 6 мы постарались избавиться от перебранки, осложнявшей список Perl 5 «портируйте это», в конце девяностых. Кое-кто говорит, что мы решили эту проблему, просто перетащив всех чокнутых в Perl 6. Мне нравится думать, что мы всех умиротворяем. Списки рассылок Perl 5 и Perl 6 пребывают в зрелом и здоровом состоянии. И заразы, пачкавшей список «портируйте это» в конце девяностых, больше нет.

LXF: Потому что всем хочется сыграть свою роль в Perl 6?

НТ: И им хочется увидеть, куда мы движемся, они знают, к кому обратиться, они знают, что к их мнению и к их замечаниям прислушаются и учтут. Тут как-то… [*молчит*] Я много думал об открытых проектах. По-моему, каждый проект Open Source сталкивается со своего рода истощением. Программисты приходят, остаются, уходят. И раньше мне казалось, что это плохо. Я помню, как я встревожился, впервые обратив внимание, что людей, изо всех сил продвигавших версию 5.004, уже не стало к релизу 5.005, они вообще не вносили своей лепты. Я думал: «Наша проблема – потребительское отношение к людям, они просто перегорают». Но потом, к 1998 году я понял, что если человек не уйдет из проекта, когда он созрел для этого – когда он уже внес посильный вклад в общее дело – то получится группа людей, которым внести просто нечего. А это куда хуже. Надо признать, что люди будут приходить, а потом уходить, и это здоровый процесс. Признак нездоровья как раз заключается в том, что никто ничего не делает, все только болтают, бьют в литавры или ворчат.

Проблема Perl 5 заключалась в том, что там, по-моему, было много «дедов», а не новичков. Open Source частенько предубежден против новичков, а нашей проблемой в Perl был перебор старичков в проекте. Я счастлив, что смог сказать: «Все, ухожу. Буду рад помочь, если потребуется, но в списке рассылки не останусь, не собираюсь лезть со своими идеями, я прекратил работу, я отошел от дел». Думаю, тогда это было нужно.

Наше сообщество сейчас куда здоровее, чем в конце девяностых. Прийти к этому было непросто, но мне нравится та энергия, которую мы видим сегодня.

LXF: Это правильно подмечено – я сам участвовал в проектах, и постепенно от них устаешь, хотя они тебе продолжают нравиться.

НТ: А худшее, что можно сделать, это продолжать навязывать свои идеи 2006 года проекту 2009. Открытый код – это разработчики, которые чешут там, где у них зудит, а если вы чешете там, где зудит не у вас, из этого ничего хорошего не выйдет, и ваши идеи будут не так интересны и своевременны, как у тех, кто чешет там, где зудит у них.

LXF: Можно, я поспрашиваю вас об OSCon, которая начиналась как конференция по Perl – много ли усилий вы в нее вкладываете?

НТ: Много. После OSCon я беру отпуск на пару месяцев, чтобы восстановить силы, но к ноябрю мы снова начинаем отправлять запросы на участие. В феврале мы отбираем доклады, к марту…

LXF: И что, каждый может выдвинуть свою идею?

НТ: Абсолютно. Это свободный сбор докладов, каждый может представить предложение по выступлению или по семинару. У меня есть программный комитет, с которым я работаю, из специалистов по Linux, и Java, и Perl, и Apache, и Python, и по прочим темам, которые мы затрагиваем, и я предоставляю им выбирать лучшие доклады и лучшие презентации.

LXF: Значит, оценку производят равные?

НТ: Как велит мода. А потом прихожу я, злобный диктатор, чтобы сказать: «Все отлично, но мне не нужно четырех семинаров на почти одинаковые темы». На семинары, после докладов, самый большой дефицит времени, и я стараюсь обеспечить здравый баланс и разумный подбор.

LXF: А сильно ли давление спонсоров?

НТ: Очень незначительно. Мы стараемся сохранять очень четкую редакторскую границу между технической программой и спонсорской стороной дел. Например, я не вписываю спонсоров, я не работаю с ними. Если они… наши спонсоры хотят включить в программу некие продукты и услуги, то с самого начала возле них появляется пометка «продукты и услуги», и это красноречивая индикация. Если вы – спонсор высшего звена, вы получите время для выступления, но я должен сперва пообщаться с докладчиком и убедиться, что доклад будет интересным, относящимся к теме и не имеющим рекламного характера. Это – очень хорошая [в действии] политика.

Я бывал на конференциях, где время, предоставленное спонсорам, было непомерно раздуто, да и у нас была пара таких случаев в самом начале. А наша политика работает намного лучше; я могу подойти к человеку и сказать: «Давайте я помогу вам найти подход к аудитории и помогу в подготовке самого выступления, чтобы ваши 15 минут доклада не превратились в 15 минут враждебности».

LXF: Корпоративные докладчики могут жутко досаждать – как заноза в пальце!

НТ: Точно. Некоторых я отправлял назад и говорил: «Не могли бы вы прислать докладчика получше? Не хочется, чтобы зрители помирали со скуки!» LXF

Читайте больше!

По мнению Ната, свободное ПО удерживает демократию в правильном русле. Читайте полную версию интервью на www.linuxformat.co.uk/torkington.html.



Firefox 2: взгляд изнутри Mozilla Foundation

Медовый месяц *Firefox 1.x* позади: браузер вернулся в куда более агрессивную среду конкуренции за любовь пользователей. Выстоит ли он? Митчелл Бейкер из Mozilla Foundation обрисовывает свой мастер-план **Грэму Моррисону**.

F*irefox* стал в своем роде феноменом. Это один из редких проектов с открытым кодом, который мы можем поддержать всей массой: здесь нет разделения на KDE или Gnome. *Firefox* преобразует Web, и впервые за много лет web-дизайнеры должны принимать в расчет что-то кроме *Explorer'а*.

Продвигает *Firefox* в массы и популяризирует его Mozilla Foundation (Фонд Mozilla). Он был создан в 2003 году как некоммерческая организация на обломках AOL'овского покровительства Netscape. AOL передал Фонду, целью которого стало обеспечение организационной, юридической и финансовой поддержки проекта *Mozilla*, интеллектуальную собственность Netscape. В определении миссии фонда есть слова: «чтобы сохранить возможность выбора и внедрять инновации в Интернете» – фраза, вполне подходящая для определения самого проекта *Firefox*.

Главный спорщик

Недавно нам выпал шанс побеседовать с Митчелл Бейкер [Mitchell Baker], руководителем *Mozilla* Foundation, о том, как их организация планирует поддерживать дальнейший рост *Firefox*, с тех пор как его основные функции (наличие вкладок, блокировка всплывающих окон и закладки RSS) освоены и другими крупными браузерами. Но сначала мы спросили, удивила ли ее популярность *Firefox*. «Да! Мы знали, что это – хороший продукт», сказала она, «но угадать, как его примут остальные, кроме нашей инициативной группы и группы разработчиков, очень сложно. Это вроде создания нужного продукта в нужное время: временем управлять нельзя, но можно ему соответствовать. В наши дни много говорят о контенте от пользователей, о влиятельных сообществах и о заразительном поведении – и мы как раз в центре всего этого».

Бейкер гордо называет свою роль в фонде Главным спорщиком (Chief Lizard Wrangler), но в обычной терминологии, она – руководитель, умело использующий свои возможности для продвижения проектов фонда. Она говорит: «За эти годы я создала процесс, и политику, и организацию, а сейчас, когда наша работа расширилась, [мне] надо просто следить, чтобы все было взаимосвязано. Мне кажется, что я озвучиваю видение проекта *Mozilla* и его индивидуальность. Я не то что являюсь его воплощением, но стараюсь ухватить все, что люди впоследствии превращают в нечто интересное».

Единства в оценках нет, но уважаемый www.webhits.de предполагает, что *Firefox* имеет 17.9% рынка браузеров, в то время как *Internet Explorer* удерживает 70.8% а *Safari* – 2.7%. Бейкер объясняет успех *Firefox* сочетанием функциональности и дизайна, но продвижение первого поколения браузера отличалось от задачи, стоящей перед Mozilla Foundation сегодня. Первая версия воспользовалась затишьем на рынке: на тот момент инновации в браузерах почти отсутствовали, и *Firefox* стал на определенное время единственным надежным продуктом, имеющим и вкладки, и блокировку всплывающих окон. Но за последние два года остальные браузеры его догнали. При отсутствии уникальных функций, дальнейший «сбыт» релиза *Firefox 2.x* потребует новой стратегии.

Свободный из свободных

Если одной функцональности мало, то как же Mozilla Foundation сохранит популярность *Firefox*?

Бейкер сообщила о планах уравновесить отсутствие крутых новых функций в новой версии *Firefox*. «Первое, что я сделаю – четко озвучу, что мы делаем и зачем. Есть продукт – *Firefox*, и это отличный продукт, намного лучше, чем прочие. Он по-прежнему лучший, но только в игру по новой вступил Microsoft, и преимущества стали менее очевидны. Если сравнивать по функциям, то, я считаю, превосходство по-прежнему за нами, но для большинства это вовсе не факт, поэтому встает вопрос: «В чем же на самом деле разница между Mozilla и Microsoft, или *Firefox* и *IE*?» Функционал – важная часть, но он – не главное».

Бейкер считает, что для продвижения Mozilla и *Firefox* нужно привлечь внимание к различию философий *Firefox* и его конкурентов (особенно Microsoft), с упором на свободу, простоту в использовании и доступность Сети для всех. Лозунг «Вернем Сеть!» (Take back the web) всегда в большей степени относился к онлайновой свободе, чем к функциональности, и Бейкер думает, что это становится все важнее.

«Я думаю, что в ближайшие полгода часть нашей работы будет заключаться в попытках сформулировать [философию] четче и громче, чтобы нашим приверженцам было проще довести ее до других, – сказала она нам. – Мы получили огромный выброс энергии, эмоций: «Вау, потрясно, этого нельзя не попробовать!». И если дать нашим людям какие-то рычаги, все будет куда эффективнее. Сообщество Spread Firefox было очень полезно – эти люди могли сказать: «Вот вам инструменты». Вопрос в том, кто мы, зачем это надо и почему мы не просто чей-то конкурент; мы все это создали, знаете ли, чтобы улучшить общение с Сетью, а не увести денежный поток от других продуктов.»

> «Лозунг «Вернем Web!» всегда имел в виду свободу, а не функциональность.»

Но и при таком подходе важно не отстранить людей, которые пользуются *Firefox* просто потому, что считают его лучшим, или делают первые шаги во Всемирной сети. Бейкер соглашается: «Мы, конечно, должны заявлять о нашей философии, это – ключевой момент. Но я понимаю, что есть люди, испытывающие дискомфорт в Интернете. Он пугает [их], потому что сложный, потому что не всегда работает, и наша роль – помочь обычному потребителю понять его и не бояться. Вот тут и нужен *Firefox*, благодаря его гибкости и способности к расширению». В качестве примера Бейкер привела расширение, созданное совместно с Joga.com, web-сайтом любителей футбола, который спонсируют Google и Nike: ваша версия *Firefox* помечается в соответствии с выбранной вами футбольной командой – участником чемпионата мира, и выводит изображения национальных флагов и популярных игроков на свободных местах своего окна (*см. экранный снимок слева внизу*). Эту расширенную версию (Joga.com Companion, www.joga.com/jogacompanion.aspx) скачали сотни тысяч раз за первые два дня после ее выхода в июне.



› Митчелл Бейкер начала работать с Интернетом в 1994 году как юрист Netscape.

«В основном об этом сообщалось существующим пользователям *Firefox*, в качестве эксперимента», пояснила Бейкер. «Следующий шаг – сообщить об аналогичной акции людям вне сообщества пользователей *Firefox*, и посмотреть, можно ли утверждать: «Да, это ценное качество, возможности расширения действительно велики, и мы умеем делать такие вещи» и показать таким образом нашу ценность».

Философские размышления

Перевод технологий «за кадр» и передача управления пользователям – основная идея широко разрекламированного Web 2.0, и развитие *Firefox* отразило такой подход к web-дизайну. Мы поинтересовались у Бейкер, считает ли она это простым совпадением, или *Firefox* сыграл свою роль в упрощении Web 2.0. «Да мы это видели и слышали отовсюду, именно этого от нас ждали», сказала она. «Мы именно и хотели попытаться привнести возможность выбора и инновации в Web. Это же чудесно, и вряд ли вы захотите лишиться этого!»

«Мы должны много работать, чтобы поддержать интерес и доверие людей к нам. Я никогда не скажу, что мы «сумели» добиться успеха – потому что перед нами еще долгая дорога – но шаги по пути к успеху нас вознаградили. И процветание Web 2.0 – это здорово!»



› Mozilla Foundation создал для Google и Nike окно браузера с темами Чемпионата мира.

»





Что еще?

Firefox усиливает позиции на домашних ПК, но на предприятиях пока виртуально невидим. Изменится ли эта ситуация?

Похоже, что насчет домашних пользователей у *Firefox* вполне солидный план действий, но если он сумеет переломить к себе отношение, то пора внедряться на офисные машины. Пусть у *Internet Explorer* есть всем известные проблемы с безопасностью – зато и известно, как с ними справиться. А вот *Firefox* ИТ-менеджеры считают неизвестной величиной, неспособной предложить тот уровень защиты и пригодности к работе, который дает Microsoft.

Понятно, что Бейкер с этим не согласна: «Знаете, *Firefox* настолько безопаснее всего используемого на предприятиях, что тут и говорить не о чем. Если сегодня провести опрос на тему «Доверили бы вы вашу внутреннюю или внешнюю сеть, да что бы то ни было, браузеру от Microsoft?», то вряд ли аргументом за использование *IE* будет его безопасность.

В самом Интернете мы обнаружили интересную вещь: если у вас [в бизнесе] есть клиенты, попадающие к вам через web, то предприятия не против создания сайтов, совместимых со стандартами, но не использующих компоненты Active X, то есть доступных не только в *IE*. А вот интранет не создает прибыли, это внутренний инструмент, скорее затратный, чем доходный. Мы видим всплеск обращений к *Firefox* по выходным, когда люди дома, и спад в течение рабочей недели».

Новая территория

Мы спросили у Бейкер, что можно сделать, чтобы изменить отношение предприятий к браузеру. «Мы не фокусировались на предприятиях по нескольким причинам, – сказала она. – Чтобы завоевать предприятия, надо иметь организацию, подобную предприятию. Мы знаем, что на предприятиях долгосрочный цикл продаж [для ПО поддержка требуется на пять и более лет], и еще имеется множество оценок и весь этот длительный процесс, через который нужно пройти. Как организация мы не очень соответствуем этому процессу. Мы добились определенных успехов среди потребителей, поэтому предпочли направить нашу энергию именно в эту область, а отсюда уже заинтересовывать и предприятия».



› *Internet Explorer* – самый популярный браузер, но после выхода Firefox 3.0 это может измениться.

Удержать равновесие, добавляя новые функции и не отпугивая при этом новых пользователей – самый важный аспект в *Firefox 2.x*, но это будет непросто. Каким же образом Mozilla Foundation учтет это в новом релизе? «Вообще-то я уже написала об этом [http://snipurl.com/xine], – сказала Бейкер. – Тот факт, что мы рассматриваем заботу о пользователях и новшества раздельно, уже сам по себе создает равновесие. Нам нужно пространство для испытания новых идей, потому что многие наши опытные пользователи просто не представляют, насколько пугающим Интернет кажется большинству. Колоссальное число людей глядит в экран монитора и даже не знает, что окно, которое они видят – это программа на их машине, а информация в нем – это данные, поступающие от специального механизма. А может, им и не надо этого знать. Вы же водите свой автомобиль, не особо в нем разбираясь? Одна группа людей – люди Web 2.0 – заинтересована в постоянном обновлении, а другая группа страшится любого изменения».

> «Мы не можем выпустить Firefox 2.0 или 2.1, а потом заявить: «Ой, там ошибка!». Митчелл Бейкер

Дабы удовлетворить ветеранов и не запугать новичков, разработчики нового релиза *Firefox 2.0* добавили функций, но скрыли их от глаз. Как объясняет Бейкер: «В *Firefox 2.0* есть функции, невидимые до тех пор, пока они не нужны – например, anti-phishing [защита от обмана] – с ними не надо разбираться, чтобы научиться пользоваться нашим продуктом; однако они проявятся. И мы рассматриваем возможность добавки функций, которые я называю «фишками». Скажем, вы смотрите в окно, но это только окно, а сам Web намного шире. Как добавить к нему «фишки» с информацией, интересной лично вам? Может быть, расширение Weather Forecasting могло бы помочь маленькой кнопкой с прогнозом погоды?».

Погоня за идеями

Mozilla интересует возможность появления «фишек», когда пользователь того пожелает, причем без ущерба основным функциям *Firefox*. С целью исследовать эту проблему и другие новшества, для их тестирования и обсуждения созданы Mozilla Labs, бывшие Mozilla Prototypes (http://wiki.mozilla.org/Mozilla_Prototypes). «У нас будет сайт и группы пользователей, и сообщество, и ряд спецпроектов, чтобы ответить на вопрос: «Какие главные инновации надо исследовать и оценить, и как их внедрить, чтобы не спугнуть те массы пользователей, которые нам

Объяснение некоторых жаргонизмов

XULRunner

XUL, XML User Interface Language (Язык пользовательского интерфейса XML) используется при разработке виджетов для web-страничек, он был создан проектом Mozilla.

XULRunner – среда исполнения для приложений XUL, то есть предназначенная для установки, распределения и удаления приложений на основе XUL, а также для обеспечения основных функций: например, криптографии, web-сервисов (используя SOAP) и сетевых соединений. Среда разработана проектом Mozilla, так что и *Firefox*, и *Thunderbird*, и *Songbird* могут использовать XULRunner, и самый первый релиз продукта будет совмещен с *Mozilla Firefox 3.0*.

SQLite

Приложения, требующие доступа к базам данных, обычно устанавливают соединение с текущим процессом либо на локальном компьютере, либо по сети. *SQLite* работает по-другому. Вместо этого, все функции базы данных вносятся в особую библиотеку, напрямую связанную с приложением, желающим ею пользоваться. Программист получает доступ к базе данных, используя вызовы функций, а не средства IPC; это значительно увеличивает эффективность. *SQLite* включен в *Firefox 2.0*, но хранение закладок и истории посещенных сайтов в БД в последний момент вырезали. Эти функции появятся в *Firefox 3.0*.



доверяют?» В этой области мы будем проводить все больше экспериментов с широкой общественностью и сообществом пользователей. Например, в *Firefox 2.0* войдет база данных *SQLite*. Но эта функция не будет кардинально менять жизнь среднего пользователя», – говорит Бейкер.

«Мы думаем, что эта программа поспособствует эксперименту: «Что интересного можно сделать?» и «Как познакомить людей с новыми функциями так, чтобы объяснить их пользу и никого не отпугнуть?» Инновациям нет конца. Но, с другой стороны, десятки миллионов людей боятся перемен – как нам подружить их с новыми полезными функциями? Нельзя же испытывать это на наших основных проектах – я имею в виду, мы не можем выпустить *Firefox 2.0* или 2.1, а потом сказать: «Ой, там ошибка».

Возможно, эта стратегия отпугнет хакеров, и они уйдут к *Konqueror* или *Nautilus*, где можно самим создавать функции, или к *Opera*, где можно развлекаться с поддержкой *BitTorrent*. Но достойно восхищения, что Mozilla отдает приоритет обычным пользователям.

Firefox остается движущей силой, и в будущем Mozilla Foundation сможет применить ту же модель развития к другим своим проектам. «Да, у нас уже давно дисбаланс в распределении ресурсов. Львиная доля была у *Firefox*, и так будет и дальше», – объясняет Бейкер. «*Firefox* – очень важный слой. Сейчас мы упаковали [технологию, стоящую за *Firefox*] в штуку под названием XULRunner – это рабочая среда для приложений, ориентированных на работу в браузере или в Web. Вообще-то, *Firefox* может существовать отдельно от XULRunner. На XULRunner сделан большой акцент, но… львиная доля отдана *Firefox*. Это великая цель – попытаться влиять на развитие Интернет в лучшую сторону и облегчить пользование им. Есть много вполне довольных пользователей *Thunderbird*, но если говорить о внедрении нового и переменах, об Ajax и новых приложениях, их продвигает *Firefox*».

Пока еще рано предсказывать для *Firefox 2.0* лидирующие позиции с самой первой версии: конкуренция обещает быть жесткой. *Firefox* начинался как «легкий» браузер, и жить или не жить его расширениям, зависело только от их популярности. Добавочные функции его не обременяли, и *Firefox 2.0* очень похож на своего предшественника: это шлифовка идеи и солидная основа для роста. И главное – следующей версии ждать недолго.

Более грандиозные планы строятся по поводу *Firefox 3.0*, намеченного к выходу в 2007 году. На тот же срок планируется выпуск операционной системы Microsoft Vista, а значит, команде разработчиков *Firefox* надо будет рискнуть. К тому времени появится новая версия движка *Gecko* и родная версия для Mac OS X; и у функций, которые не вошли в версию *2.0* – например, «Places» и хранения журнала посещений в БД SQL – будет время дозреть.

Правила могут меняться, но *Firefox* с самого начала создавался с возможностью адаптации. И это – большее, чем можно сказать о любом из его конкурентов. LXF

> **«Может, эта стратегия и обратит хакеров к другим продуктам, но достойно восхищения, что приоритет отдан обычным пользователям.»**

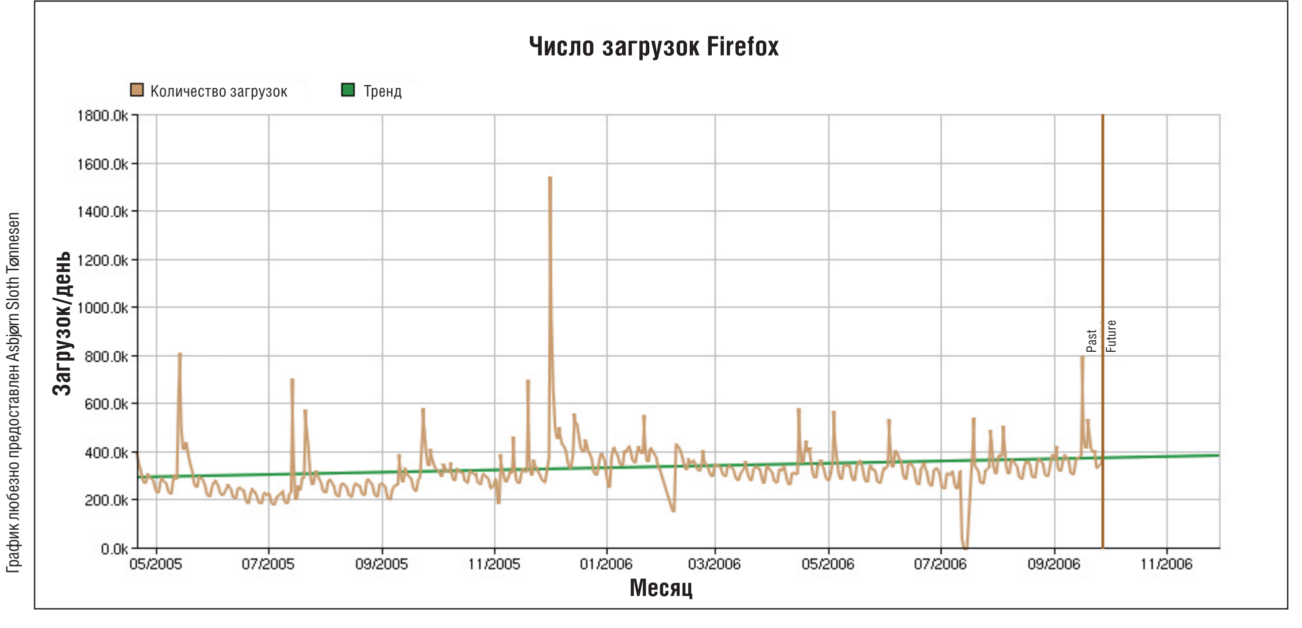


График любезно предоставлен Asbjørn Sloth Tønnesen

Небольшие пики в течение каждого двухмесячного периода соответствуют всплеску интереса к Firefox в выходные, отмеченному Бейкер.

Что за штука....



Хорошие новости для тех, кто скачивает много информации, а также пользователей VoIP: **Дэвид Кулсон** расскажет о технологии, помогающей интернет-пакетам дойти до получателя.

Хм, DCCP. Похоже на DCC из *IRC*. Это для обмена файлами?

На самом деле, из общего у DCC [Direct Client-Client, Прямое соединение клиент-клиент] и DCCP [Datagtam Congestion Control Protocol, Протокол управления прохождением датаграмм] только буквы D и C в названиях. DCCP – совсем другой зверь, он помогает приложениям, занимающимся рассылкой файлов, а не их обменом.

Вы сказали «прохождение» – что, DCCP ускоряет Интернет?

Нет, но увеличивает надежность доставки данных. Сегодня большая часть интернет-трафика лежит на плечах TCP или UDP. TCP используется для долговременных соединений типа SSH, HTTP и FTP, которым присущи регулирование нагрузки и гарантия доставки данных (как часть протокола). UDP используется, в основном, для протоколов с коротким временем работы, включая DNS. А поскольку UDP – это протокол, не сильно озабоченный вопросом потери пакетов, он применятся для вещания аудио и видео, включая VoIP.

Значит, UDP быстрее, чем TCP, но не гарантирует надежности передачи?

По-простому, да. Есть сервисы, требующие данные сразу же, причем данные должны быть свободны от ошибок. Однако ошибки бывают всегда, и DCCP призван добавить в UDP устойчивость к сбоям протокола TCP, без необходимости переписывать приложения, чтобы учесть новую специфику доставки данных.

DCCP – отличная идея. Почему его не используют вместо TCP и UDP?

TCP и UDP появились раньше, чем сам Интернет, и для практических целей они будут работать еще долго. TCP идеально подходит для основательной и надежной передачи данных, прямо как во время сессии Telnet, когда код нажатой клавиши гарантировано передается через сеть. UDP, напротив, полезен как протокол типа «послал и забыл».

Это не здорово.

Смотрите: когда мы делаем запрос DNS, то посылаем запрос на удаленный сервер и какое-то время ждем. Если ответ не получен, мы просто работаем дальше, а если получен несколькими секундами позже, мы его игнорируем – отсюда и название «послал и забыл». Использование протокола TCP для такого типа запросов неэффективно, так как перед передачей данных требуется всякий раз устанавливать соединение.

DCCP явно станет штатным протоколом вместо UDP, например, в RTP, где возможность реакции на потерю или затор пакетов, увеличивающий латентность, увеличит надежность протокола в целом.

Латентность...?

Задержка передачи пакета данных.

А, да. Как все сложно...

Вовсе нет.

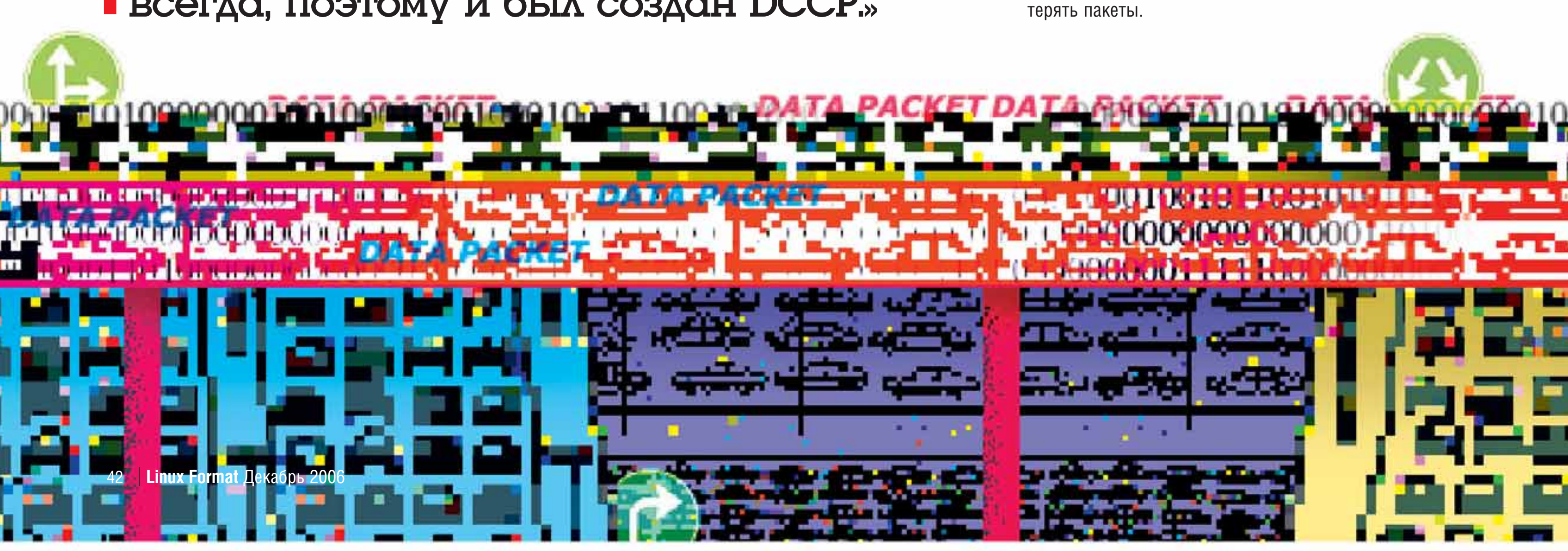
Так как же DCCP улучшит UDP?

Сессия UDP не имеет состояний. Клиент посылает пакет серверу, а сервер посылает пакет обратно. Не выполняется ни инициализация соединения, ни его завершение после передачи данных. В отличие от UDP, DCCP создает соединения – точнее, «полу-соединения» (half-connections), которые можно рассматривать как ненадежные однонаправленные каналы (pipes). Посылка данных по такому соединению похожа на посылку писем – нет гарантии, что они дойдут до адресата, а если и дойдут, то вряд ли будут получены в том же порядке, в каком отправлялись. Многие протоколы на основе DCCP будут использовать пару односторонних соединений для передачи данных между узлами сети, а заодно передавать подтверждения получения пакетов.

Именно этого UDP не делает.

Верно. Как TCP, DCCP позволит клиенту повторять посылку пакетов, если сервер не подтвердил их получение. Один из результатов такого подхода – возможность ограничивать темп передачи данных и поднимать реальную скорость передачи до такой, которую позволяет пропускная способность, и при этом не терять пакеты.

> «Есть сервисы, требующие данные сразу и без ошибок. Однако ошибки бывают всегда, поэтому и был создан DCCP.»



» Почему посылка большого объема данных вызовет потерю пакетов?

Любой канал в Интернете имеет определенную ширину, и данные передаются надежно, только если их объем не превышает ширины канала. Когда канал загружен максимально, передадутся не все данные. Интернет-устройства, включая обычные персональные компьютеры, способны ставить данные в очередь, и если пропускной способности канала недостаточно для немедленной отсылки, то устройства подождут, а потом отошлют данные в том порядке, в котором они помещались в буфер. Но если и буфер полон, и канал полностью загружен, то пакеты будут теряться.

» Теперь понимаю.

TCP – довольно умный протокол, и когда сталкивается с подобной ситуацией, то просто уменьшает скорость передачи данных, чтобы гарантировать доставку пакетов.

» А как DCCP ограничивает трафик?

DCCP отличается от TCP тем, что управление нагрузкой инициируется из процесса, находящегося в пространстве пользователя, который и создал одностороннее соединение, а не находится «под колпаком» у ядра. На текущий момент существует два профиля для управления нагрузкой, известные как CCID2 и CCID3. CCID2 похож на TCP, он тоже быстро реагирует на флуктуации трафика в Интернете, позволяет соединению использовать всю доступную полосу пропускания и препятствует потере пакетов. CCID3 пытается поддерживать определенный уровень передачи данных и избегает быстрой адаптации, используя процесс, известный как дружественное к TCP управление скоростью, или TFRC.

» Какие интернет-сервисы будут использовать CCID2 или CCID3?

CCID2 подходит тем приложениям, где поток передаваемых данных не постоянен, например, интернет-играм или в качестве замены традиционного UDP, и позволяет устранить задержку, ускоряя доставку пакетов. CCID3 идеален для приложений, посылающих данные равномерно, например, VoIP или других медиа-потоков, потому что таким приложениям важнее поддерживать постоянное прохождение данных между узлами, чем максимально использовать полосу пропускания канала.

> «Многие интернет-провайдеры уже ограничивают протоколы на базе UDP, часто применяемые для DoS-атак.»

» Как убедить приложения перейти с UDP на DCCP?

Для конкретного приложения мгновенной выгоды может и не быть. Но протоколы используют пропускную способность канала все больше, и интернет-провайдерам придется задуматься над выбором между протоколами, сжирающими ресурсы каналов, и протоколами, помогающими уменьшить нагрузку. Многие провайдеры уже ограничивают использование протоколов на базе UDP, часто применяемых для совершения DoS-атак (Denial of Service, способ вывести сервер из строя, забросав его неподъемным количеством запросов) или других потенциально вредоносных процессов; в результате, страдают VoIP и другие современные медиа-технологии.

» UDP может вызвать DoS-атаки?

Именно что может. Из-за того, что не инициируется соединение, система может послать UDP пакет на удаленный узел с подставным адресом отправителя, а сервер отошлет пакет на подставной адрес отправителя в качестве ответа. Таким образом можно предпринять довольно эффективную распределенную DoS-атаку, особенно если сервер генерирует ответный пакет большего размера по сравнению с полученным запросом.

» Есть ли поддержка DCCP в Linux?

Начиная с Linux 2.6.14, поддержка DCCP включена в основное дерево исходного кода ядра. Как обычно, потребуется некоторое время, прежде чем производители популярных дистрибутивов включат в свои продукты необходимые патчи для других компонентов, включая Netfilter, для полной поддержки DCCP.

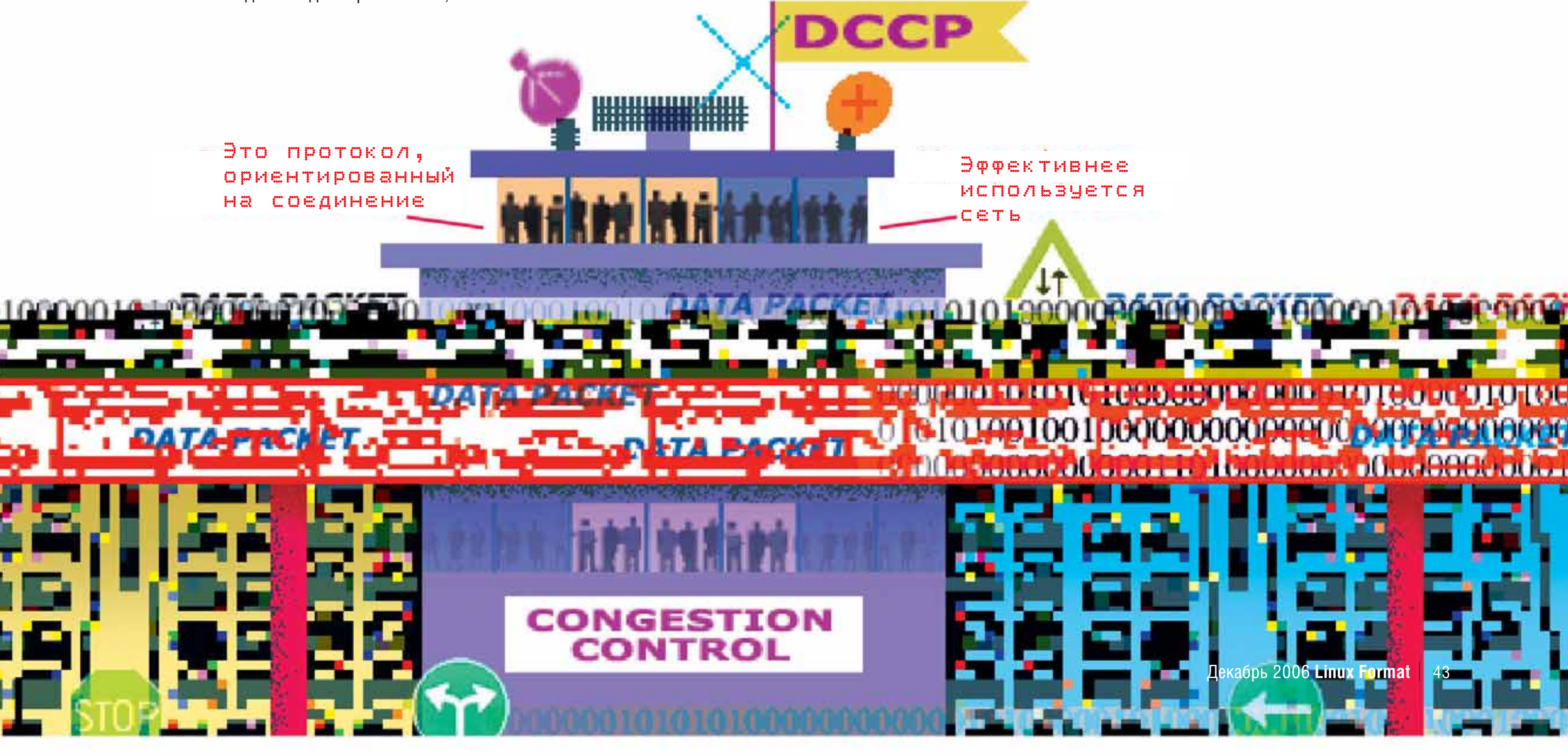
» И как вы думаете, когда приложения начнут поддерживать DCCP?

Уже доступно некоторое число тестовых приложений для оценки DCCP-протокола в Интернете и внутри локальной сети. Документация по DCCP находится на http://linux-net.osdl.org/index.php/DCCP. Ruby и Python поддерживают DCCP, библиотеки Perl уже могут использовать возможности DCCP, так что почти любая программа, скрипт или приложение могут использовать новый протокол.

Поддержка DCCP сейчас включается в репозитории Tcpdump и Etherape, что обеспечивает поддержку всех текущих IP-протоколов, которые используются в Интернете. Очевидно, необходимость в этом есть, так как для адекватного использования нового протокола необходимы инструменты для поиска неисправностей и отладки.

» Спасибо, вы меня очень просветили. А не посоветуете ли какой-нибудь сайт?

Не за что. Подробности вы получите на www.read.cs.ucla.edu/dccp. LXF



А ваш бизнес в безопасности?

Сбой!

Потеря информации!

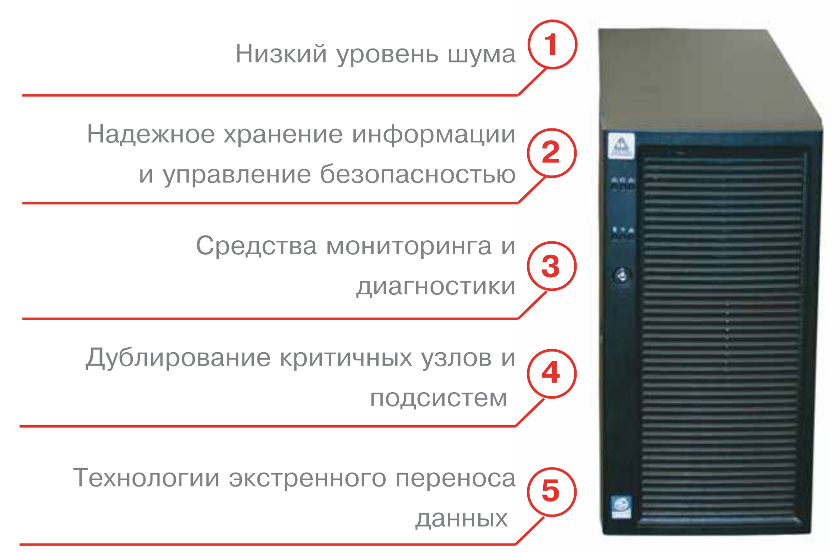
Остановка бизнес-процессов!

Xeon inside

Включи режим максимальной защиты данных

Сервер DESTEN Navigator DX 8000L

на базе процессора Dual-Core Intel® Xeon®



Низкая совокупная стоимость владения достигается за счет поддержки отрытых архитектур и использования современных средств администрирования, программного обеспечения и пакета услуг по развертыванию и поддержанию работоспособности системы.

Расширенное сервисное обслуживание:

- Предоставление подменного оборудования
- Быстрое время реагирования (до 2х часов)
- Круглосуточная горячая линия техподдержки

Процессоры Dual-Core Intel® Xeon® 5000 серии
Память до 32GB двухканальная FBDIMM (8 слотов)
Сеть интегрированы 2 сетевых адаптера Gigabit Ethernet на базе Intel 82563EB
HDD до 10 SATA или 10 SAS с "горячей заменой"

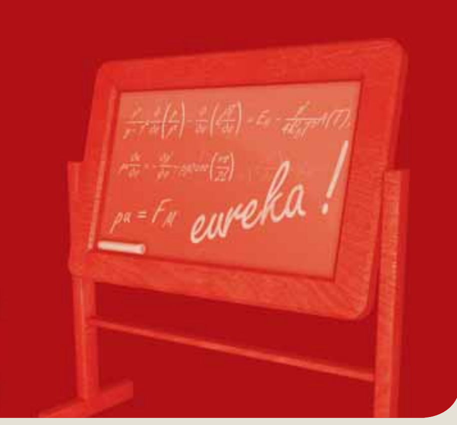
Белгород, «Оверсан», ул. Садовая, д. 45а, (0722) 26-29-01, 31-02-83, 26-19-41 / **Благовещенск**, Амурская область, «Эстел», ул. Зейская, д. 173А, (4162) 53-40-30, 51-40-30, 53-41-37 / **Волоколамск, МО,** «ТОРИС», ул. Сергачева, д. 18/7, оф. 2., (496) 362-4067 / **Ижевск**, «Десten», ул. Воткинское шоссе, д. 140, т.: (3412) 44-34-00, тел./факс 46-04-23 /**Лабытнанги**, Тюменская обл., Ямало-Ненецкий АО, «Ямал КЦ», ул. Школьная, д. 20, т.: (34992)-23332, Москва – (495)602-34-16 / **Лабытнанги**, Тюменская обл., Ямало-Ненецкий АО, «Ямал КЦ», ул.Гагарина, д. 24, т.: (34992) 23-332 / **Липецк**, «Сетевые технологии», ул.Студеновская, д. 3, д.: (0742) 47-99-77 / **Липецк**, «Империя», ул.Студеновская, д. 3, т.: (0742) 47-99-77 / **Магнитогорск**, «Верисел-сервис», пр-т К. Маркса, д. 50, т.: (3519) 22-64-15, 22-78-49 / **Москва**, «Информационные Банковские Системы. Консалтинг», ул. Киевская, д. 21, оф. 7, т.: (495) 240-74-67, 240-73-43, 240-79-13 / **Мурманск**, «Сервис центр ТИС», ул.Папанина, д. 47, т.: (8152) 42-09-09, 42-48-07, 42-48-08, 42-48-09 / **Нерюнгри**, «Компьютерный центр «Десten», пр. Дружбы Народов, д. 29/1, т.: (41147) 4-34-54, 4-45-15 / **Новокузнецк**, «СОТЧИ-net», ул. Кирова, д. 64, т.: (3843) 35-28-78 / **Новокузнецк**, «СОТЧИ-net», ул. Дружбы, д. 39-230, т.: (3843) 35-28-78 / **Ноябрьск**, «Мегабайт», ул. Энтузиастов, д. 22, л.: (34564) 1-01-73/74 / **Орел**, «Квант», ул. МОПРа, д. 12, т.: (0862)75-24-29, 75-24-30, 47-15-09, 75-24-29, 75-24-30, 47-15-09 / **Протвино, МО**, «Гармония Про», ул. Ленина, д. 18, оф. 198, т.: (27) 74-26-22 / **Санкт-Петербург**, «DESTEN Computers», ул. Большая Подъяческая, д. 35, пом. 7Н, т.: (812) 310-02-76, 570-29-69 / **Челябинск**, «Контур», ул. Постышева, д. 6, кв. 32, (351) 264-98-99, 263-47-88 / **Южно-Сахалинск**, «Меридиан», ул. Хабаровская, д. 2, т.: (4242) 42-40-53, 42-36-73 / **Южно-Сахалинск**, «Компьютеры и Связь», ул. Ленина, д. 213, оф. 114, т.: (4242) 74-49-47, 74-44-62

Реклама

Единая информационная служба: +7 (495) 970-00-07
м. «Киевская», Бережковская наб., 20
www.desten.ru



LINUX FORMAT



Учебники

Наши эксперты помогут вам с любым приложением Linux



ЕВГЕНИЙ БАЛДИН
Начинал с Агатов. Когда-то даже знал, что такое Робик.

Знаковые слухи

Революция свершилась прямо на наших глазах: Java выйдет под GPL2, а известный производитель альтернативной операционной системы фирма Microsoft прикупила кусочек Linux, заключив соглашение с Novell. Теперь осталось понять, к чему бы это.

Можно по-разному относиться к Java, но это действительно промышленная технология. В нее «вбухана» прорва денег, человекочасов и даже идей. Собственно говоря, у Sun Microsystems особого выбора не было, потому что промышленная технология живет, когда ее принимает сообщество. А сообществу, похоже, уже смекнуло, что свобода для инструмента — это одна из наиболее ценных особенностей. В Sun впечатлились и сделали шаг в правильном направлении.

Порадовал один из основных поставщиков новостей для LOR – фирма Microsoft. Novell, правда, жалко. Но очевидно другое – Linux интересен. Только выражается это интерес как-то завуалированно: «Novell and Microsoft collaborate – customers win». Неубедительно. Возможно, их «проймет» чуть попозже.

Ведь Sun уже проняло — мало им Java, они теперь и OpenSolaris подумывают отпустить под GPL. А то Linux удушил многие из коммерческие Unix-ов. Ему нужен спарринг-партнер посерьезнее, которого задушить действительно невозможно.

Пусть в честной борьбе победит достойнейший, и это не станет причиной для поражения.

E.M.Baldin@inp.nsk.su

В этом выпуске...

Совет месяца: Специальные клавиши

Многие из нас имеют клавиатуры с дополнительными клавишами, назначение которых ясно следует из их маркировки. Проблема в том, что лишь немногие из ныне существующих дистрибутивов способны корректно настроить их. В итоге вы жмете – а ничего не происходит. Более того, даже если вы захотите назначить этим клавишам какие-либо специальные функции вручную, то обнаружите, что это невозможно – они не опознаются системой.

Проблема имеет два решения. Первое – установить правильный клавиатурный профиль в **/ect/xorg.conf**. Список доступных вариантов можно найти в файле xorg.lst, который обычно расположен в каталоге **/etc/X11/xkb/rules**. Он содержит строки вида «модель описание», например:

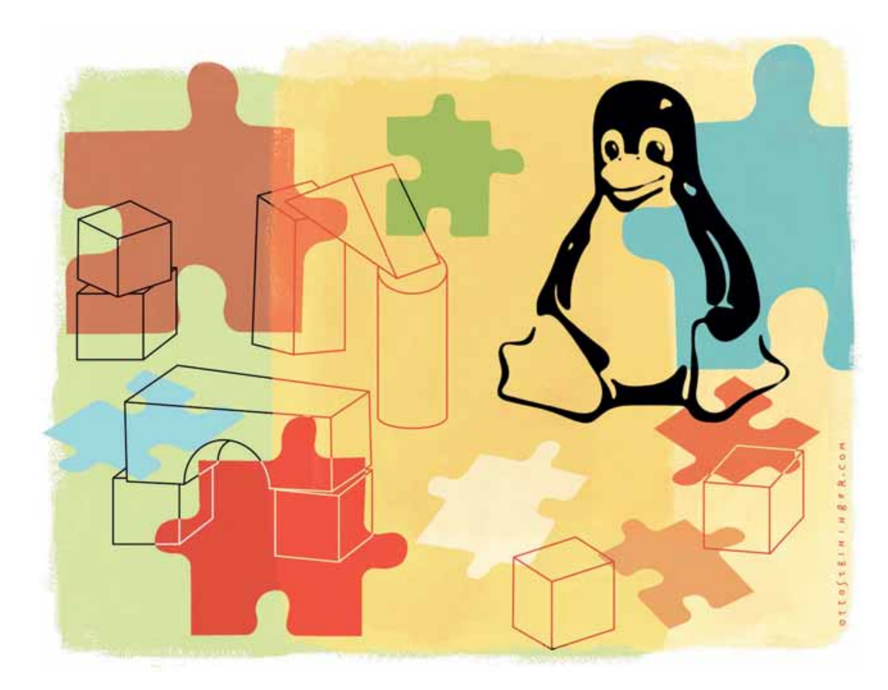
```
cherryblue Cherry Blue Line CyBo@rd
```

Если вам посчастливилось найти здесь свою клавиатуру, вставьте название модели в поле XkbModel в файле **xorg.conf** и перезапустите X-сервер. После этого вы сможете назначить специальным клавишам любые функции. Ура!

Второй (и единственно возможный в случае, если ваша клавиатура не перечислена в **xorg.lst**) способ – настроить все вручную. Здесь вам пригодится утилита *xev*, которая выводит на экран значения raw-кодов, соответствующих той или иной нажатой клавише. Запустите *xev* из командной строки и «потопчитесь» по спецкнопкам. Затем найдите в выводе *xev* слово keycode и запишите следующее за ними число. После этого создайте файл **~/.Xmodmap** и добавьте в него строку вроде: keycode 68 = F13, естественно, подставив вместо 68 значение, полученное от *xev*. Не забудьте выполнить команду xmodmap ~/.Xmodmap, чтобы сделанные изменения вступили в силу.

Wine: Windows

Программное обеспечение не станет свободным за одну ночь, и пока что приходится запускать Windows-приложения в Linux – **Энди Ченнел** покажет, как это сделать.



Свобода – это здорово, но иногда нужно срочно сделать работу, а угрызения совести оставить на потом. Ради *Photoshop*, *PowerPoint* или *Grand Theft Auto* многие законопослушные линуксоиды тайком обращаются к своим Windows-разделам, не найдя в Linux достойного аналога.

Но зачем использовать Windows-приложения непременно под Windows? Одна из альтернатив – установка *Wine*, свободной системы совместимости. Она дублирует многие функции Windows API и способна запустить немало родных приложений Windows. Не нужно дополнительно обзаводиться Windows или запускать полноценную гостевую ОС, чтобы просто поработать с небольшой программой. К сожалению, *Wine* иногда не справляется с запуском конкретного приложения – хотя по мере взросления проекта это встречается все реже и реже (вы также можете взглянуть на список альтернативных вариантов на стр. 48).

На данном уроке мы установим *Wine* и посмотрим, как здесь воссоздается файловая система Windows для запуска Windows-приложений. Опробуем также одну из наиболее популярный реализаций *Wine* – *CrossOver*, позволяющую работать с пакетом *MS Office*.



Наш эксперт

Энди Ченнел
Энди делает свои первые шаги в Linux уже шесть лет, а технологиями интересуется еще со времен Dragon 32.

Часть 1 Установка Wine

Для данного урока я использовал простую, но обновленную установку Ubuntu, хотя если *Wine* уже получен и установлен, то процедура инсталляции и запуска приложений будет одинакова в Fedora Core, Linspire или любой другом дистрибутиве. Как и для других программ Linux, имеется несколько способов установки *Wine*. В порядке убывания сложности, это сборка из исходных текстов, загрузка и установка пакета с www.Winehq.com и использование менеджера пакетов вашего дистрибутива. Мы собираемся воспользоваться последним способом, но не потому, что я такой глупый – просто тогда приложение включается в систему обновления дистрибутива, и вам всегда будет доступна новая версия.



› **Wine доступен через Synaptic, менеджере пакетов Ubuntu.**

Выбрав способ установки, можно двигаться дальше. В Ubuntu это означает запуск *Synaptic* (Система > Администрирование > Менеджер пакетов Synaptic) и использование инструмента Поиска для обнаружения *Wine*, который должен быть доступен в виде отдельного пакета. Пометьте его для установки и нажмите Применить для загрузки и установки.

Теперь откройте терминал и наберите **wine** для автоматической настройки. Если вы желаете убедиться, что *Wine* установлен, откройте ваш домашний каталог, выполните Вид > Показывать скрытые файлы и продвиньтесь вниз до **.wine** (напомним, что Linux скрывает файлы и каталоги, начинающиеся с точки). Заглянув в этот каталог, вы обнаружите пару директорий с именами **Program Files** и **Windows**; это аналоги их тезок в реальной системе Windows. Я всегда добавляю каталог **Downloads** (стандартно: правый щелчок и затем Создать > Каталог) как репозитарий для загружаемых установочных файлов. На то есть две причины: во-первых, я аккуратист, а во-вторых, это облегчает поиск приложений, которые необходимо переустановить – они всегда в одном месте.

Хочу предостеречь вас: вы могли подумать, что *Wine* – эмулятор Windows... а вот и нет. Это API-совместимая прослойка, и назвав ее эмулятором в присутствии разработчиков *Wine*, вы станете белой вороной.

» **Месяц назад** Мы добавили запросы, отчеты и виджеты в базу данных *OOo Base*.

под Linux!



Часть 2 Запуск Windows-программ

Теперь переберемся в терминал. Я сам патологически боюсь работать с компьютером без графических инструментов, но в конце концов это себя оправдывает – и мы не засядем там надолго, обещаю!

Откройте терминал – *Konsole*, *Gnome Terminal*, вообще любой – и наберите **winecfg** для запуска утилиты настройки приложения. А не сработает, попробуйте скомандовать **/usr/local/bin/winecfg**. Запустится простенькая утилитка, которую можно использовать для указания версии Windows, которую вы хотите, кхе, эмулировать, и различных настроек: звука, видео и прочего. Скорее всего, вам ничего не придется менять здесь, но правка файла настройки приложения может пригодиться, если, например, окажется, что оно разработано для Windows 95 или отказывается работать в режиме более чем 8-битного цвета.

Что для нас важнее, в этой утилите также указывается расположение различных «вместилищ файлов», которыми пользуется Windows, например, Мои документы и Мои рисунки. По умолчанию, они попадают во всегдашнюю папку **/home/имя_пользователя**, но ее легко заменить (используя соглашения об именах файлов в Linux) любой другой, чтобы приложения Windows правильно интегрировались в рабочий стол Linux. Для изменения этих настроек, выберите запись Мое нечто, нажмите кнопку Выбрать и задайте местоположение по вашему выбору.

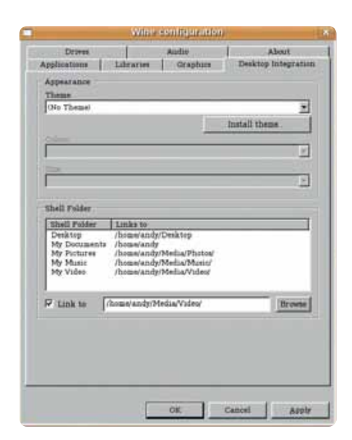
Итак, *Wine* установлен и настроен, пришло время скогтить какое-нибудь приложение. В моем случае это небольшой, бесплатный (как пиво) персональный финансовый менеджер под названием *AceMoney Lite* (www.mechcad.net/products/acemoney/index_lite.shtml). Это не только отличный финансовый пакет, но и Windows-приложение, дружественное к *Wine*. После загрузки приложения переместите его в корень файловой системы лже-Windows. Linux видит его как директорию:

```
/home/andy/.wine/drive_c/
```

Windows (а значит, и *Wine*), однако, считают, что это

```
c:\\
```

Это различие становится важным, когда мы начинаем устанавливать Windows-приложения.

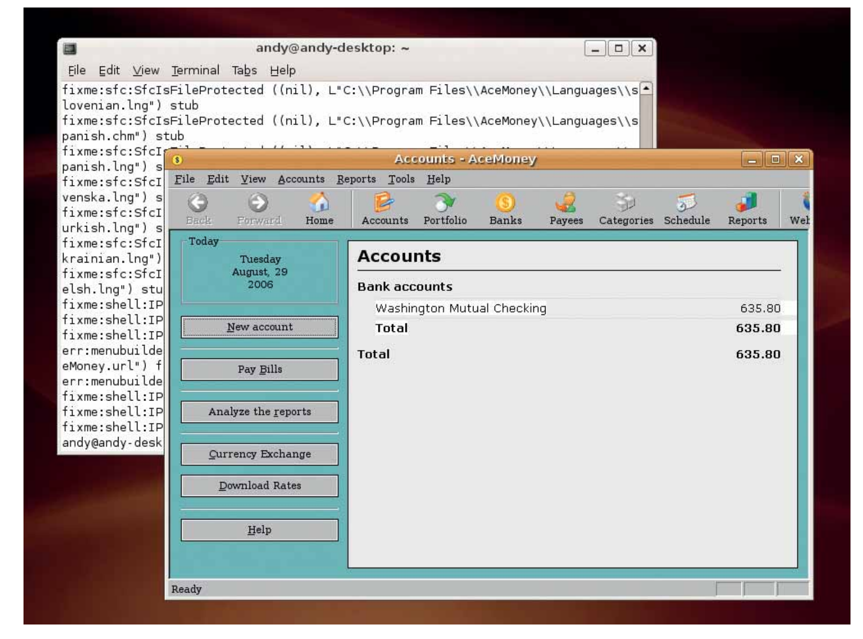


› Производите настройку так, чтобы приложения попали в должные места Linux.

Как и большинство других Linux приложений, *Wine* можно запустить, открыв терминал, набрав имя приложения и нажав клавишу Enter. Разница в том, что мы должны предоставить *Wine* топливо для работы, а именно, путь к запускаемому приложению. Выполните

```
wine c:\\AceMoneyLiteSetup.exe
```

Вторую часть этой команды замените на имя приложения, которое выбрали вы. Приложение должно запуститься (если *Wine* умеет с ним работать), а остальная процедура установки аналогична таковой в Windows.



› Установив приложение в *Wine*, считайте его родным Linux-приложением.

Ярлыки на рабочем столе

После успешной установки приложения, вы можете запустить его, вновь открыв терминал и набрав

```
wine c:\\path\\to\ the\\application.exe
```

Обратите внимание на важные особенности записи этого пути. В Windows для указания подкаталога используется один обратный слэш (\), но оболочка Linux трактует его как признак спецсимвола, поэтому используется двойной обратный слэш (\\). Необходимо также «экранировать» слэшами пробелы, так что путь Windows вроде **C:\Program Files\MyApp\MyApp.exe** превращается в **c:\\Program\ Files\\MyApp\\MyApp.exe**.

Как сказано выше, я ужас как не люблю пользоваться командной строкой, когда можно обойтись щелчком мыши – поэтому добавлю к нашей команде симпатичную иконку и помещу ее на рабочий стол. Щелкните правой кнопкой мыши в любом месте рабочего стола и выберите Создать > Ссылка на приложение. Введите имя приложения (если хотите, снабдите его описанием), затем – команду запуска приложения. Теперь нажмите кнопку со стандартной иконкой, выберите из приведенных иконок то, что вам понравится, и нажмите OK. Дважды щелкните на новом ярлыке, и voila! Процесс Windows-программы пошел.

Чтоб жизнь малиной не казалась, метод добавления *Wine*-записей в меню приложений Gnome слегка отличается от добавления ярлыка на рабочий стол. В Ubuntu структура меню редактируется в редакторе *Alacarte*, доступном через Applications > Accessories. Я собираюсь создать отдельный раздел меню с именем «Приложения Windows»; выполните Файл > Новое меню, дайте ему имя и нажмите OK. Стрелками у правого края окна *Alacarte* можно пользоваться для перемещения вновь созданного раздела меню вверх и вниз по списку. »



Установите ли вы Wine или CrossOver Linux, все равно программы будут припрятаны в скрытых каталогах (обозначенных именем, начинающимся с точки) и станут видны, только если вы сами сделаете их видимыми, выбрав Показывать скрытые файлы в вашем файловом менеджере.

Скорая помощь

Поскольку не все установки проходят гладко, я рекомендую иметь в вашей системе и *Wine*, и *CrossOver*. При этом, если *CrossOver* проваливает установку (как это произошло с *AceMoney Lite*), то, скажем, вы можете воспользоваться *Wine*, и наоборот.

» Выбрав в главной панели пункт Приложения Windows (Windows Applications), следует нажать Файл > Новый и затем ввести детали в диалоговом окне. Вот и большое отличие: вместо предыдущего метода ввода пути следует использовать метод Linux, а именно указать команду (**wine**) и полный путь (**/.wine/drive_c/Program Files/AceMoney/AceMoney.exe**), заключенный в одиночные кавычки. Выглядит это так:

```
wine '/.wine/drive_c/Program Files/AceMoney/AceMoney.exe'
```

Сделав это, добавьте подходящую иконку, убедитесь, что небольшая кнопка слева от записи Приложения Windows нажата (чтоб сделать меню видимым в главном меню), и нажмите Файл > Выход или кнопку Закрыть. Теперь вы можете запустить программу, нажав Приложения > Приложения Windows > AceMoney.

Все работает, и это хорошо, но упрощение действий – еще лучше; и мы в состоянии сделать работу в Linux прямо-таки магией. Поэтому настроим рабочий стол на установку исполняемых EXE-файлов Windows просто по двойному щелчку на иконке. К счастью, это просто, как дважды два, и лишь слегка отличается от стандартного метода создания «файловой привязки». Сначала найдите установочный EXE-файл, потом щелкните на нем правой кнопкой.



› **Любители простора на рабочем столе Ubuntu предпочтут создать пункты меню для своих Windows приложений.**

После этого выберите пункт меню Открыть в другом приложении... *Wine* обычно работает в фоновом режиме, поэтому не имеет собственного пункта меню, и не по чему щелкнуть для его выбора в качестве соответствующей программы. Вместо этого надо щелкнуть на маленькой иконке внизу окна с именем Использовать Другую Команду; появится поле ввода текста. Наберите в нем **wine**, затем нажмите Открыть. Выбранный EXE файл запустится через *Wine*, и начнется установка. А поскольку для данного типа файлов других привязок-ассоциаций нет, то эта станет стандартной.

Ну вот и все. Список поддерживаемых приложений и другая информация находится на http://appdb.winehq.org. *Wine* и все его производные – это полумера, полезная, пока большая часть Windows-разработчиков не прозреет и не портирует свои приложения в Linux. Однако эти полумеры работают, и неплохо, и по здравом размышлении вы сможете навсегда проститься со своим разделом Windows.



› ***Wine* легко назначить стандартным приложением для EXE-файлов.**

Различные варианты Wine

Wine – проект популярный, и к нему имеется немало дополнительных инструментов, улучшающих качество установки и управления приложениями Windows в Linux.

» CrossOver Linux

Стандартный *Wine* справляется с тоннами приложений, но для установки некоторых больших программ, типа *MS Office* или подключаемых модулей для браузеров вроде *QuickTime* и *Shockwave*, рекомендую приобрести копию *CrossOver Linux* (бывший *CrossOver Office*) от *CodeWeavers* ($39.95, на сайте www.codeweavers.com). Это не только упростит управление приложениями Windows, но и поддержит разработчиков, делающих популярные программы доступными в вашей любимой системе.

CrossOver предоставляет более устойчивое окружение для приложений, чем *Wine*, и упрощает их интеграцию с хост-окружением. Например, создание привязок для файлов (скажем, открытие OFX-файлов при помощи *AceMoney*) при помощи *Wine* может быть трудным, а CrossOver имеет для этого специальный раздел.

» Cedega

Этот коммерческий продукт от TransGaming Technologies (цена от $15 за трехмесячную подписку на www.transgaming.com) позаботился об игроках. Как *CrossOver* Linux оптимизирован под конкретное промышленное ПО, так *Cedega* оптимизирован под игры. Число поддерживаемых игр растет с каждым обновлением, среди них *World of Warcraft*, *Battlefield 2*, *Civilization IV* и *Half Life 2: Episode 1*.

» WineTools

Это приложение хорошо уживается с *Internet Explorer 6*, *Photoshop 7.0*, *Illustrator 9.0* и многими другими, но устанавливает свою версию *Wine*, вместо той, что уже имеется. Программа доступна на сайте www.von-thadden.de/Joachim/wineTools/index.html в виде RPM или статически скомпонованного двоичного файла.

» WineDoors

Пользователи Gnome, страдающие по родному графическому клиенту для управления *Wine*-установками, должны дождаться выпуска этого приложения, хотя и ранние версии выглядят вполне достойно. *WineDoors* создается в *Eclipse*, и текущая версия доступна только через *SVN*. Вероятно, лучше не устанавливать его сейчас, а подождать появления пакетов.

» WineXS

Это еще один проект, начатый Фрэнком Хендриксоном [Frank Hendrikson], отцом *WineTools*. Интерфейс пользователя очень прост и предоставляет доступ к реестру *Wine*, инструменту настройки *Winecfg* и набору других сервисов «в одно нажатие». Есть также приличный инструмент установки, и благодаря использованию текущей системной инсталляции *Wine* вы избавлены от необходимости помнить, где какое приложение находится – это проблема *WineTools*.

Ни одно из этих приложений не свободно от ошибок. Некоторые программы работают в *Wine*, но не работают в *CrossOver*, и наоборот. К счастью, о конфликтах беспокоиться не нужно. Каждая реализация *Wine* создает личные каталоги лже-Windows. Например, что-то из *CrossOver* вы можете найти в **/home** в подкаталоге **.cxoffice**; а *Wine* хранит всё своё в подкаталоге **.wine**.

» Wine@Etersoft

Отечественная разработка, позволяющая запускать в Linux популярные российские программы: *1С: Предприятие 7.7*, *1С:Бухгалтерия 6.0*, *Консультант Плюс*, *Инфо-Бухгалтер*, *Кодекс*, *Референт*, *ДубльГИС*, программы подготовки обязательной отчётности, а также *MS Office 97* и некоторые другие. специалисты компании Etersoft тесно сотрудничают с основным проектом *Wine* и обмениваются наиболее критичными патчами, но сам продукт *Wine@Etersoft* – коммерческий, его распространением на территории РФ занимается LinuxCenter.Ru.



› ***Cedega* поддерживает многие Windows-игры.**

Шаг за шагом: неподдерживаемые приложения в CrossOver



1 Поддерживаемые программы

Они могут быть загружены непосредственно из меню. В главном окне выберите Install (Установить), затем выберите приложение из списка. Оно будет установлено и добавлено в меню, как и следует.



2 Неподдерживаемые случаи

Выберите Install Unsupported Software (Установить неподдерживаемую программу). Откроется новое окно для выбора файла. Нажмите Other *.exe Location (Расположение другого *.exe). Такой выбор объясняется использованием загруженного приложения.



3 Ищем EXE-файл

Выбор третьей опции запустит менеджер файлов *CrossOver*, весьма похожий на окно выбора файла в Windows. Перейдите к месту расположения вашего EXE-файла.



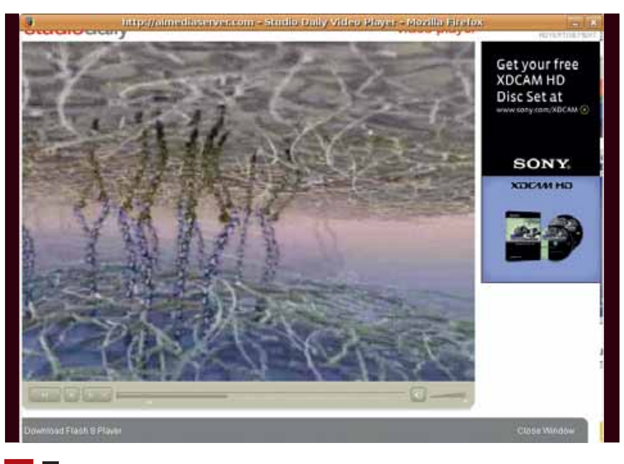
4 Убедимся, что видим приложение

Установщик сам заполняет меню Windows Applications, а вручную это делается на вкладке Меню: выберите нужное приложение и нажмите кнопку Добавить Пункт Меню (Add Menu Entry).



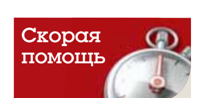
5 Берем Flash Player 9 для Windows

При установленном *Wine* добавьте Microsoft Core Fonts (Базовые шрифты Microsoft) командой **sudo apt-get install msttcorefonts**. Загрузите последнюю Windows-версию Firefox с www.mozilla.org и сохраните ее, как и раньше, на **c_drive**.



6 Теперь установим программу

Дважды щелкните на нем в каталоге и введите **wine .wine/drive_c/Program\ Files/Mozilla\ Firefox/firefox.exe**. На www.adobe.com щелкните по Get Adobe Flash Player и установите EXE-файл. LXF

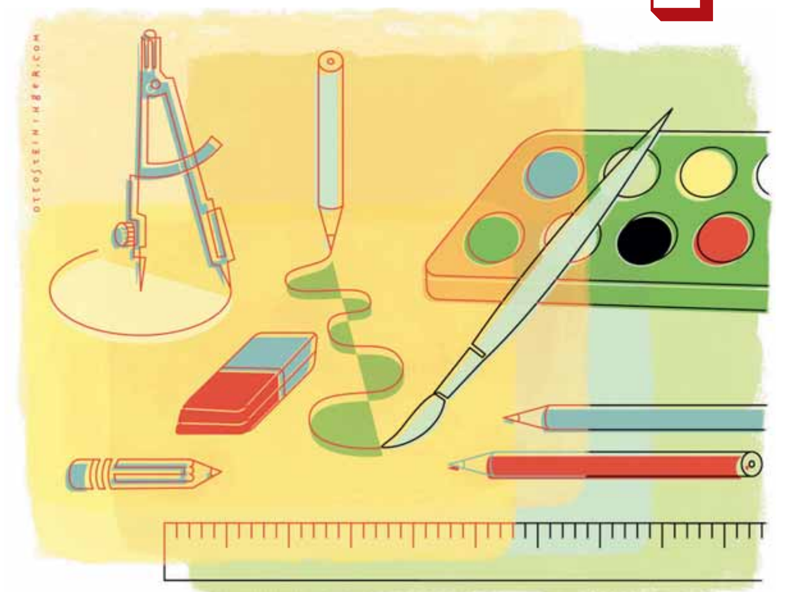


Будьте осторожны: некоторые EXE-файлы на самом деле являются самораспаковывающимися архивами и могут заполонить ваши каталоги DLL- и CAB- файлами, а также файлами readme. Кроме того, известно, что Wine весьма эффективно запускает вирусы Windows.

Через месяц Как использовать сканер для любых целей в системе Linux.



Inkscape: Тест



ЧАСТЬ 6: В чем работать – в *Inkscape* или в *Xara*? Расследует **Дмитрий Кирсанов**.

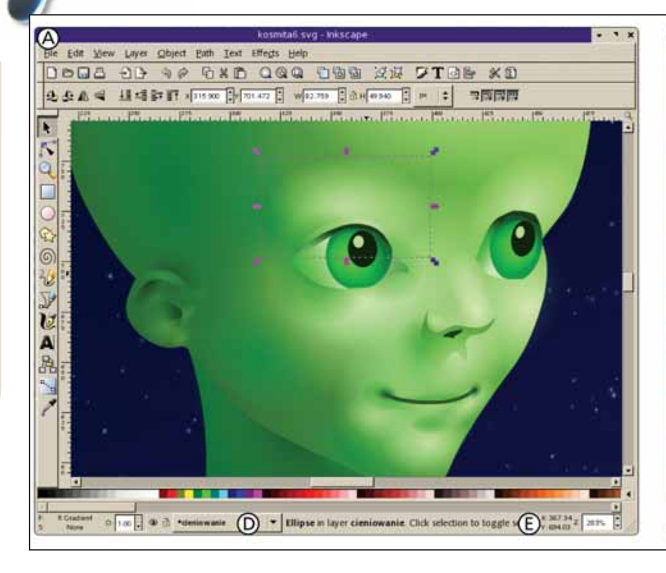
Это случилось в октябре 2005. *Inkscape* только-только утвердился в качестве самого многообещающего векторного инструмента для Linux, уже пригодного для работы над реальными проектами, и быстро набирал пользователей и разработчиков: многим открылся мощный потенциал векторного редактора с открытым исходным кодом.

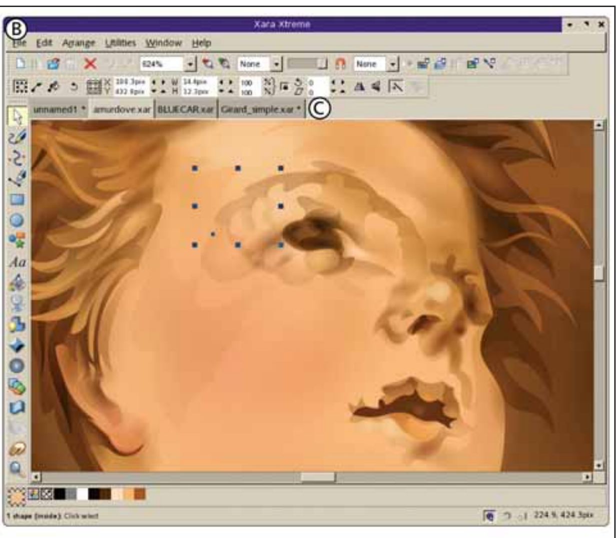
И тут на поле вышел новый игрок. Небольшая, но влиятельная английская фирма *Xara* (www.xaraextreme.org) объявила, что опубликует исходный код своего флагманского продукта, векторного редактора *Xara Xtreme*, и портирует его в Linux. Не являясь промышленным стандартом, программа тем не менее была вполне солидна и уважаема еще с восьмидесятых (сперва на Atari, потом под Windows) и имела внушительное сообщество преданных пользователей.

Наш эксперт

Дмитрий Кирсанов Графический дизайнер, специализирующийся на создании логотипов и web-сайтов. Консультант, пишет книги и статьи о дизайне и XML-технологиях.

Шаг за шагом: Чья возьмет?





1 Организация окна

С первого взгляда на *Inkscape* (A) и *Xara* (B) заметно базовое сходство обеих программ. Меню, кнопки под ним, главная панель инструментов слева, цветовая палитра и строка состояния внизу – эти компоненты одинаковы. У *Xara* больше инструментов в панели, а *Inkscape* обладает более развитой системой меню. Горизонтальная панель прямо над окном (называемая Info bar в *Xara* и Tool Controls bar в *Inkscape*) имеет одинаковое назначение в обоих приложениях: здесь собраны органы управления текущим инструментом. Со сменой инструмента содержимое панели меняется.

Но есть и различия. Благодаря долгой истории коммерческого применения, интерфейс *Xara* компактнее (меньше размер шрифта), проработаннее и чище. (Можно подумать, что столь мелкий шрифт непрактичен, но, судя по опыту профессионалов, кучность дизайнеры ценят выше читаемости). *Xara* обладает многими удобствами, которых недостает *Inkscape*, например, открытием документов во вкладках (C) и плавающими панелями инструментов – попробуйте и убедитесь. С другой стороны, *Inkscape* щеголяет удобным селектором слоев справа от строки состояния (D), и панель сообщений (E) здесь более информативна, чем в *Xara*.

Месяц назад: Мы испытывали ключевые функции *Inkscape 0.44* в славном *LXFicolor*!

на фоне Xara

Одной из причин, побудивших *Xara* сделать подобный выпад, был стремительный прогресс *Inkscape* благодаря новым, уникальным функциям. С другой стороны, *Xara* для разработчиков *Inkscape* всегда была примером продуманности интерфейса и отменной практичности. И хотя обе программ давно знали друг о друге, сейчас они впервые встретились как соперники.

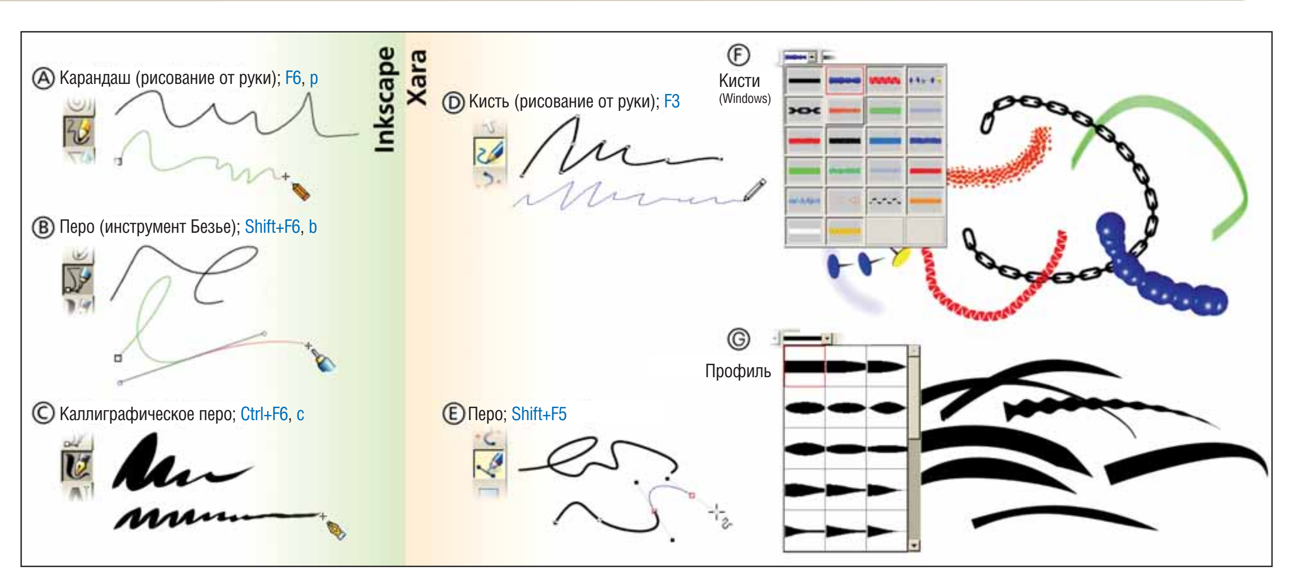
Соперники? Да, иначе не охарактеризуешь нынешние взаимоотношения этих проектов. Оба объявили себя открытыми и выразили готовность обмениваться идеями и кодом; ходят даже слухи о грядущем объединении, правда, обусловленные недопониманием технической сложности объединения столь непохожих программ. А разнятся они во всем, начиная с геометрических алгоритмов, продолжая несовместимыми форматами файлов (базирующийся на XML SVG в *Inkscape* и собственный двоичный формат XAR в *Xara*) и заканчивая разными интерфейсными библиотеками [это не совсем так. *Inkscape* использует *GTK*, *Xara* – *wxWidgets*, который «транслируется» в *GTK* под Linux, – прим. ред.].

Хотя вывод новой Linux-версии (*Xara Xtreme* for Linux) на один уровень с Windows-редакцией – первоочередная задача, вариант 0.7 пока уступает Windows-версии в функциональности. И все же сравнивать две программы уже можно. За этот короткий урок я проведу базовое сравнение; надеюсь, оно позволит вам получить представление о каждом проекте и поможет выбрать для себя подходящий инструмент.

На каждом шаге мы будем сравнивать различные аспекты программ, а граница между «зонами» *Inkscape* и *Xara* будет сдвигаться влево или вправо в зависимости от развитости соответствующего инструмента в рассматриваемой области. Экранные снимки сделаны с новейших версий каждой из программ, находящихся в разработке (*Inkscape 0.44+*, *Xara 0.7+*), хотя иногда я пользовался Windows-версией *Xara*, чтобы продемонстрировать то, что еще недоступно на Linux.



«Обе программы «знали» друг о друге, но сейчас они впервые встретились как соперники.»



2 Основы рисования

Давайте начнем с чистого холста и испробуем инструменты для рисования той и другой программы. У *Inkscape* три таких инструмента: Pencil (Карандаш) для простых набросков от руки (A); Pen (Перо) для точных прямых и кривых Безье (B); и Каллиграфическое перо для художественного рисования, имитация каллиграфического пера (C). У *Xara* таких инструментов только два: в одном (D) объединены простые линии и художественные кисти, а второй (E) – прямой аналог Пера.

Налицо различие подходов программ к художественному рисованию. Любая линия в *Xara* может быть выполнена разными «кистями» (F) – по сути, узорами, повторяющимися вдоль линии – и иметь разные «профили», определяющие внешний вид штриха [толщину или насыщенность цвета штриха – в зависимости от выбранной кисти – при смене его направления или при изменении силы нажатия, если вы работаете с планшетом – прим. ред.](G). Несмотря на изощренность этой системы (можно, например, задать собственную кисть), ей не хватает непосредственности Каллиграфического пера от *Inkscape*. Судя по моему опыту, кисти *Xara* удобны для продуманных работ (например, рисования полупрозрачными штрихами в виде эллиптических градиентных мазков), но для быстрых набросков и рисования «для души» нет ничего лучше каллиграфического пера *Inkscape*, с его отзывчивостью, плавностью и естественностью. Кроме того, изображение с большим количеством художественных мазков может замедлить отрисовку, особенно при большом увеличении (хотя, вообще-то, рендеринг в *Xara* заметно быстрее, чем в *Inkscape*). »

»

Ⓐ Узлы; F2, n
Ⓑ Узловая пластика
Alt+drag
Alt+drag
Ⓒ Трансформация выбранных узлов
[,] поворот
<, > масштабирование
Ⓓ Выравнивание и распределение выбранных узлов
Inkscape
Xara
Ⓔ Редактор форм
Ⓕ Координаты левого манипулятора
Координаты узла
Координаты правого манипулятора
выбранный узел
Ⓖ Сглаживание выбранных узлов
0%
50%
100%

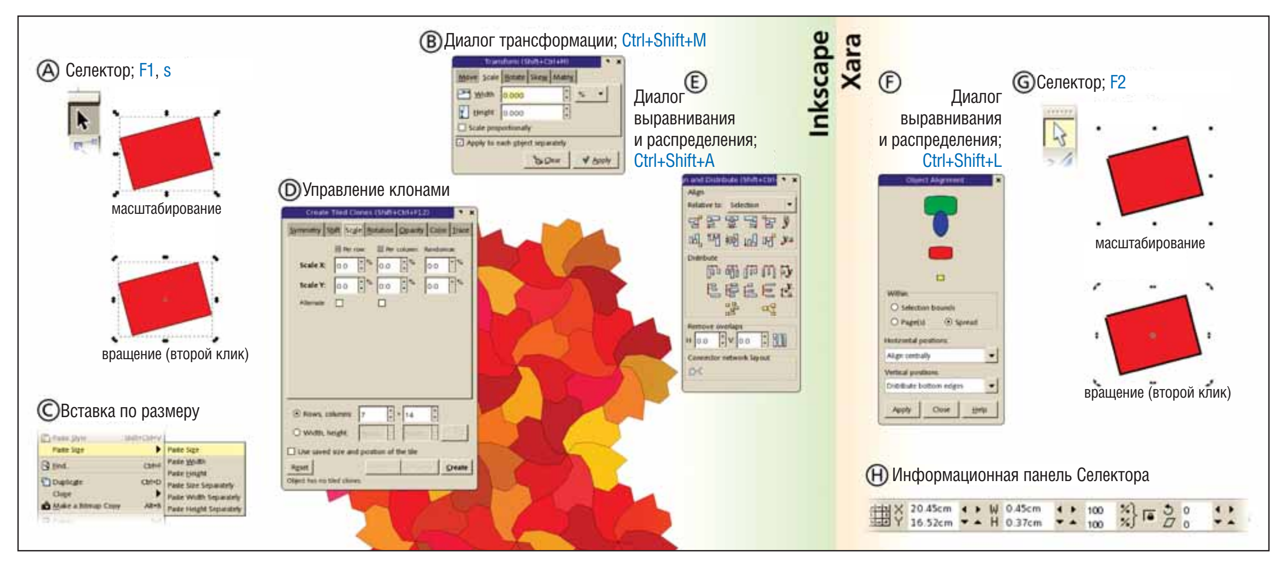
3 Редактирование узлов

Инструменты для работы с узлами, напротив, почти не имеют различий в подходах. Они называются Node (Узел, A) и Shape Editor (Редактор фигур, E) в *Inkscape* и *Xara* соответственно (хотя оба могут редактировать как линии, так и фигуры), и если вы знакомы с одним из них, другому обучиться недолго. Однако каждая программа обладает некоторыми функциями, отсутствующими у другой, что мы сейчас и увидим.

Xara умеет отображать местонахождение позиций выделенных точек и их рукояток в числовом виде (F), и упрощать (сглаживать) выделенную часть линии при помощи ползунка (G) (*Inkscape* упрощает только линию целиком по Ctrl+L, хотя иногда похожего эффекта можно добиться удалением выделенных точек). Еще одно преимущество *Xara* – здесь можно редактировать несколько линий одновременно, тогда как *Inkscape* ограничивает вас единственной выделенной линией.

С другой стороны, *Inkscape* обзавелся такими мощными функциями, как пластика узлов (B), масштабирование и поворот выделенных узлов (C). Можно даже выравнивать и распределять узлы на линии с помощью специального диалогового окна (D). Узлы также снабжены типами (кроме гладких и узлов перегиба, имеются еще и симметричные); больше вариантов соединения, разделения и копирования узлов, и значительно больше клавиатурных комбинаций – можно, например, двигать узел вдоль его рукоятки по Ctrl+Alt.

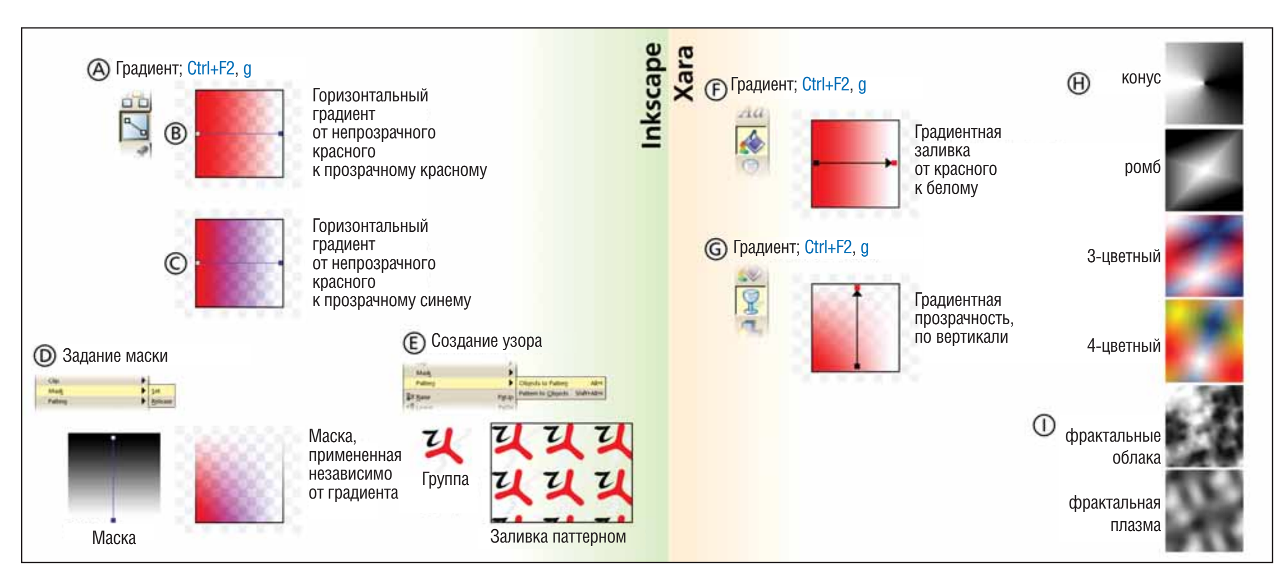
Когда вы освоите эти функции, работать с узлами в *Inkscape* станет быстрее и удобнее, чем в *Xara*.



5 Размещение и преобразования

Основные операции инструмента выделения, да и клавиатурные комбинации для преобразований, в обеих программах так похожи, что случайный пользователь может не заметить разницы (A, G). Второй щелчок на выделенном элементе в обеих программах приводит к активации рукояток вращения. Семантика команд перемещения также подобна: например, стрелки без Alt перемещают выделенный элемент на фиксированное расстояние, а нажатие Alt со стрелкой вызывает передвижение на один экранный пиксел (истинное расстояние зависит от масштаба). Однако, опытным пользователям не хватает в *Xara* горячих клавиш поворота и масштабирования ([] и <>, соответственно, в *Inkscape*). Вообще, я считаю значительным преимуществом *Inkscape* более широкий выбор клавиатурных комбинаций и возможность переопределять многие из них (предусмотрена даже эмуляция клавиатурного профиля *Xara*).

В *Xara* также недостает мощного диалогового окна преобразований *Inkscape* (B), хотя некоторые похожие функции (вращение и сдвиг) доступны через панель свойств инструмента выделения (H). Команда *Inkscape* дополнила диалоговое окно преобразований флажком Apply To Each Object Separately (Применить отдельно к каждому объекту) (B). Диалоговые окна выравнивания и распределения объектов также в основном похожи (E, F), и здесь преимущество *Inkscape* еще более очевидно: выравнивание строчек текста, рандомизация, разделение, удаление перекрытий – все эти функции уникальны в *Inkscape*, как и группа команд Paste Size (Вставка по размеру, C). К тому же *Xara* не умеет автоматически обновлять клоны, а следовательно, там нечего противопоставить диалоговому окну управления клонами (D).

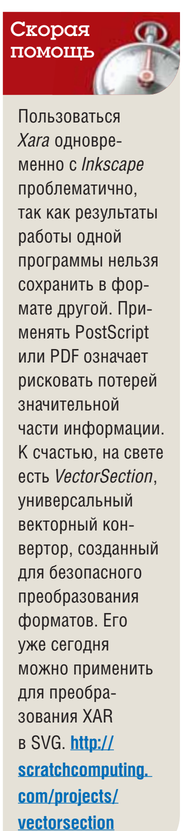


4 Прозрачность и градиенты

Обе программы способны создавать полупрозрачные объекты и градиенты. Однако и здесь налицо концептуальные различия. *Xara* использует для управления прозрачностью и для заливки два разных инструмента (F, G). Заливка цветом (помимо прочих свойств) может быть как однотонной, так и с градиентом, но прозрачность цвета неуправляема. Если требуется прозрачность (ровная или с переходом), ее нужно добавлять отдельно, поверх заливки, инструментом прозрачности (Transparency). Такой подход бывает удобен – иногда (например, при добавлении вертикального прозрачного градиента к горизонтальному цветовому градиенту или к растровому изображению), но в большинстве случаев это не так.

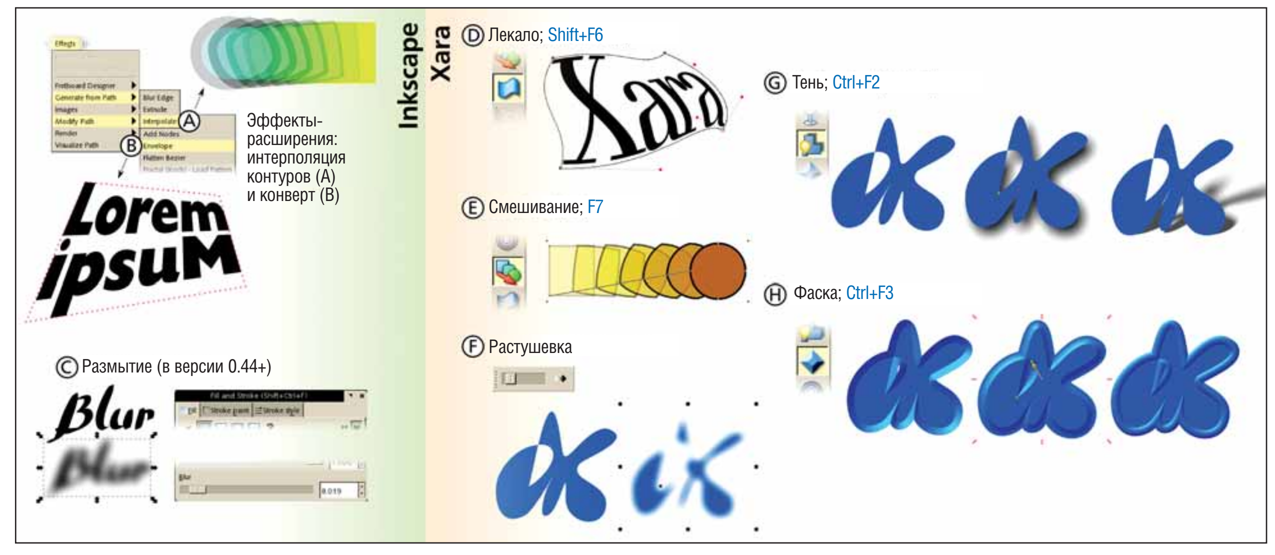
У *Inkscape* другой и, по мне, потенциально более мощный подход. Здесь применяется один инструмент (A), но создаваемый им градиент может быть прозрачным или полупрозрачным в любой степени. Простой градиент от красного до прозрачно-красного (B) легко создать в любой программе, а вот переход от красного до прозрачно-синего (C), безусловно, проще произвести в *Inkscape*. А если необходимо управлять яркостью независимо от заливки, в *Inkscape* это можно сделать с помощью масок (D), хотя и через весьма скудный диалоговый интерфейс.

У *Xara* шире выбор типов градиента – конический, ромбовидный, трех- и четырехцветный (H), тогда как *Inkscape* может предложить лишь линейный да радиальный – все, что поддерживает SVG. Еще у *Xara* есть два вида фрактальной заливки и прозрачности (I). С другой стороны, *Inkscape* может создать узорную заливку из любого объекта или даже группы объектов (E), а *Xara* заливает объекты только растровыми паттернами.



Скорая помощь

Пользоваться *Xara* одновременно с *Inkscape* проблематично, так как результаты работы одной программы нельзя сохранить в формате другой. Применять PostScript или PDF означает рисковать потерей значительной части информации. К счастью, на свете есть *VectorSection*, универсальный векторный конвертор, созданный для безопасного преобразования форматов. Его уже сегодня можно применить для преобразования XAR в SVG. http://scratchcomputing.com/projects/vectorsection



6 Эффекты и фильтры

Итак, *Inkscape* «выиграл» шаг № 5, но нельзя не упомянуть и о том, что и *Xara* обладает хорошо развитыми способностями, имеющимися в *Inkscape*, в лучшем случае, в виде примитивных расширений. Среди них инструменты Mould (Лекало) (D) и Blend (Смеситель) (E) с широким набором органов управления и удобств. Все, что *Inkscape* может им противопоставить – это два расширения («Интерполяция» и «Конверт», A и B соответственно), крайне ограниченные и лишенные гибкости – например, конверт может быть только прямолинейным, тогда как в *Xara* доступны криволинейные очертания.

Есть у *Xara* и некоторые инструменты, не имеющие прямых аналогов в *Inkscape*, которые придают обычным творениям особый шик. Таков, например, инструмент Bevel (Фаска, H), создающий трехмерные рамки с изменяемой высотой и различной направленностью освещения. Но самый популярный инструмент, вероятно, Shadow (Тень, G), создающий естественно размытые тени разных видов. Поверх всего этого с помощью панели инструментов Feather (Растушевка, F) можно применить дозированную растушевку (которая в основе своей имеет маску с приглушенной яркостью) к любому объекту. До недавнего времени *Inkscape* просто нечего было противопоставить всем этим качествам. Теперь же его тестовые версии получили инструмент Blur (Размытие, C), который делает возможным как создание теней, так и лессировку (хотя, конечно, не в один щелчок, как в *Xara*), наряду с другими интересными эффектами.

Итак, мы сравнили программы с различных точек зрения. Решение принимать, конечно, вам, но я надеюсь, что наш урок поможет оценить их возможности. Надеюсь также, что вы освоите обе программы. Обратная связь – на адрес: letters@linuxformat.ru. LXF

Настоящая безопасность Как воспользоваться умными утилитами Linux и защитить вашу машину.

Безопасность: Нет

ЧАСТЬ 3: Урок сканирования на уязвимости. **Крис Браун** покажет, как тестировать сеть на окна и черные ходы со всей серьезностью.





Наш эксперт

Д-р Крис Браун независимый инструктор по Linux, имеет степень доктора наук по физике элементарных частиц, сертифицированный специалист Novell и Red Hat. Недавно написал книгу о SUSE для издательства O'Reilly.

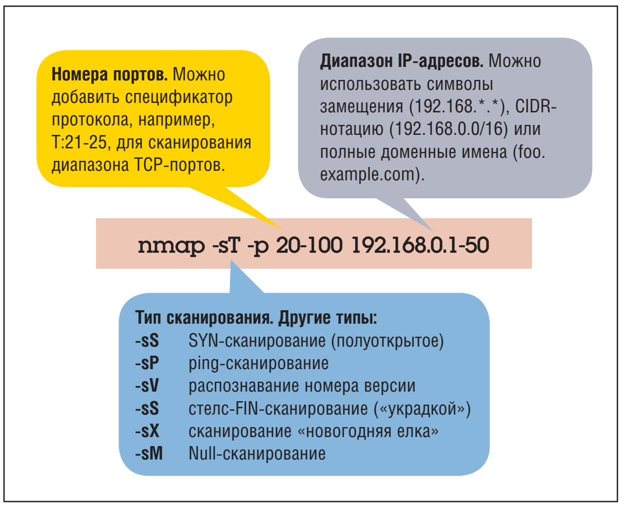
На сей раз мы рассмотрим утилиты, помогающие в поиске уязвимостей вашей системы. Очевидно, что искать уязвимости можно как с честными, так и с дурными намерениями, посему повторю то, что сказал на первом нашем уроке: во-первых, я абсолютно не поощряю использование этих утилит для получения неавторизованного доступа. Во-вторых, перед запуском этих утилит на работе вы должны получить разрешение у вашего начальника [то же самое относится и к домашним сетям, принадлежащим интернет-провайдерам, – прим. ред.].

Утилиты для оценки уязвимостей делятся на две категории: одни действуют снаружи, а другие внутри системы. Наша первая утилита, *Nmap*, явно относится к первым. *Nmap* определяет, какие порты открыты (то есть ждут соединений), и может сканировать сразу несколько числа машин: посылает серию сетевых пакетов на указанные диапазоны портов и IP-адресов и смотрит, что происходит.

Вот простой пример запуска *Nmap*. Моя домашняя сеть довольно мала – на маршрутизаторе осталось всего два целых порта, остальные погибли во время грозы – но этот вывод даст вам пищу для ума:

```
$ nmap -sT -p 20-100 192.168.0.1-50
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-01
15:18 BST
Interesting ports on 192.168.0.1:
(The 78 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
23/tcp filtered telnet
53/tcp open domain
80/tcp open http
Interesting ports on 192.168.0.3:
(The 78 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
22/tcp open ssh
25/tcp open smtp
80/tcp open http
All 81 scanned ports on 192.168.0.4 are: closed
Nmap finished: 50 IP addresses (3 hosts up) scanned in 12.269
seconds
```

Рис. 1 (ниже) показывает некоторые опции *Nmap*. Интереснее всего типы сканирования. Чтобы вполне оценить их, необходимо глубокое понимание работы TCP/IP; в частности, TCP-флагов и стандартной последовательности TCP-пакетов при создании TCP-соединения. Тип сканирования **-sT**, показанный на схеме, использует т.н. «трехстороннее рукопожатие» (three-way handshake) для установки соединения, как обычный клиент. Такое сканирование очень «заметно»: скорее всего, информация о нем попадет в файл журнала сканируемой системы. Зато его можно выполнить не от имени суперпользователя. Есть и другие, «менее заметные» типы сканирования, например, тот, что в *Nmap* называется стелс-FIN-сканированием [stealth FIN scan], когда посылается TCP-пакет с одним лишь установленным флагом FIN (это один из флагов заголовка пакета). При нормальных условиях такой пакет посылается только при закрытии соединения – и никогда не посылается до установления соединения с портом. Ответ операционной систе-



(Рис. 1) Опции *Nmap* – краткое руководство по составлению команды сканирования с помощью *Nmap* (о типах сканирования см. man-страницу).

Мы отключили ненужные сервисы, познакомились с *Bastille* и мастером безопасности SUSE.

ли у вас дыр?



мы на такие попытки поможет определить, какие порты действительно открыты и ждут соединения. Спецификации TCP/IP не всегда четко определяют поведение ОС в нестандартных ситуациях, а если и определяют, реализации не всегда соответствуют спецификациям. По ответу на нестандартный TCP-пакет можно распознать ОС. Типы сканирования посредством заведомо неверных TCP-пакетов требуют открытия raw-сокета для явного формирования заголовка, и сканирование при этом возможно только от имени суперпользователя.

Определяем ОС с помощью Nmap

Опция **-A** у *Nmap* включает функцию определения сервисов и версии ОС. Вот пример:

```
$ nmap -A scanme.nmap.org
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-
04 09:30 BST
Interesting ports on scanme.nmap.org (205.217.153.62):
(The 1657 ports scanned but not shown below are in state:
filtered)
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 4.3 (protocol 2.0)
25/tcp closed smtp
53/tcp open domain
70/tcp closed gopher
80/tcp open http Apache httpd 2.2.2 ((Fedora))
113/tcp closed auth
Nmap finished: 1 IP address (1 host up) scanned in 94.760 seconds
```

Nmap – отличный инструмент для поиска доступных в данный момент портов и сервисов вашей сети. Вы можете удостовериться, что порты, которые должны быть открыты, действительно открыты, а те, что должны быть закрыты – закрыты, и что брандмауэр ведет себя как должно. А хотите – запустите *Nmap* снаружи и внутри системы, закрытой брандмауэром, и сравните результаты (о создании брандмауэров под Linux поговорим позже). *Nmap* содержит немало трюков кроме уже описанных, например, умеет разбивать IP-пакеты на мелкие кусочки. Смысл разбиения в том, чтобы затруднить работу межсетевых экранов и систем обнаружения вторжений. Для получения более подробной информации об *Nmap* прочтите man-страницу (она на редкость хороша) или посетите http://insecure.org/nmap.

Уязвимость vs эксплойт

Мы используем термины «уязвимость» и «эксплойт» так часто, что они могут показаться взаимозаменяемыми – но это не так. Уязвимость – ошибка проектирования, позволяющая пользователю повысить свои привилегии в системе (для этого есть красивое слово: «эскалация прав»). Уязвимости – не то же самое, что ошибки конфигурации (хотя и те, и другие создают дыры в безопасности), и описанные мной утилиты не найдут ошибок типа предоставления каждому первому доступа к **/etc/shadow**.

Эксплойт – некий код, содержащий хитроумно подобранные данные или последовательность команд, предназначенный для эксплуатации уязвимости с целью расширить свои права: либо для получения контроля над атакуемой системой, либо для вывода ее из строя. Существуют «теоретические» уязвимости, т.е. такие, к которым пока не изобрели эксплойта, но эксплойтов без уязвимости не бывает.

Как Nmap говорит с вами

Nmap обычно различает одно из трех состояний порта:
- **Open** – открыт и ждет соединений;
- **Closed** – доступен, но не ждет соединений;
- **Filtered** – недоступен (возможно, запросы *Nmap* фильтруются брандмауэром), и *Nmap* не может определить, открыт порт или закрыт.

Сканирование на уязвимости

Ну вот, мы дошли до сканеров уязвимостей. Они обычно используют какую-нибудь внешнюю базу уязвимостей. Следовательно, их эффективность зависит не только от качества самого сканера, но и от того, насколько активно обновляется база. Один из лучших и активно поддерживаемых сканеров – *Nessus* от Tenable Network Security (www.nessus.org).

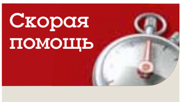
Nessus имеет клиентский и серверный компоненты. Сервер по имени *nessusd* – это часть, выполняющая сами проверки; клиент, *Nessus client*, является графической утилитой, позволяющей вам соединяться с сервером, выбирать нужные тесты и смотреть результаты. Клиент-серверная архитектура позволяет размещать сервер в разных стратегических точках сети (например, до и после брандмауэра) и проводить тесты различного назначения.

Для этого урока я установил *Nessus* на мой компьютер с Fedora Core 5. На www.nessus.org/download вы найдете собранные пакеты *Nessus* для различных дистрибутивов Linux, включая Debian 3.1, Red Hat Enterprise Linux 3/4, Fedora Core 4/5, SUSE 9.3/10 (сканер бесплатен, но исходные тексты закрыты). Я установил пакеты **nessus-3.0.3-fc5.i386.rpm** и **nessusClient-1.0.0.RC5-fc5.i386.rpm**.

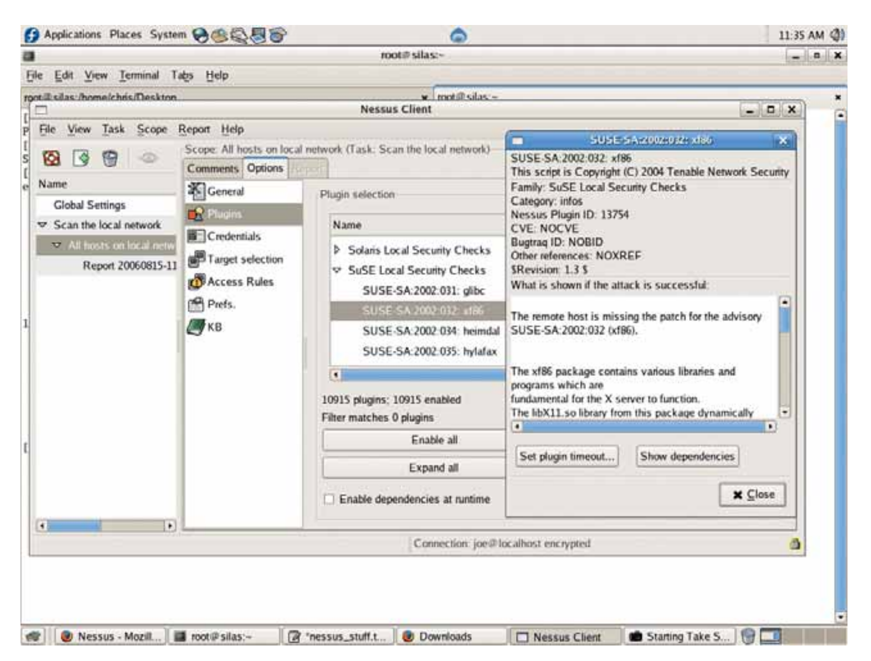
При соединении с сервером клиент должен указать верные имя пользователя и пароль, так что надо создать учетную запись *Nessus* на той машине, где планируется запуск сервера. Эти учетные записи используются в *Nessus* для контроля и не связаны с обычными учетными записями Linux. Утилита создания учетной записи *Nessus* довольно разговорчива, и использование ее самоочевидно. Вот пример диалога с ней:

```
# /opt/nessus/sbin/nessus-add-first-user
Using /var/tmp as a temporary file holder
Add a new nessusd user
----------------------
Login : joe
Authentication (pass/cert) [pass] : pass
Login password :
Login password (again) :
User rules
----------
nessusd has a rules system which allows you to restrict the hosts
that joe has the right to test. For instance, you may want
him to be able to scan his own host only.
Please see the nessus-adduser(8) man page for the rules syntax
Enter the rules for this user, and hit ctrl-D once you are done :
(the user can have an empty rules set)
accept 192.168.0.0/24
default deny
```

»



Если хотите сканировать скрытно, без ведома системы-объекта, запускайте *Nmap* от имени root.



› (Рис. 2) Экран *Nessus* для отображения подробной информации об имеющихся опциях. Его можно использовать для выбора целей, просмотра отчетов и т.д.

```
Login          : joe
Password          : ***********
DN          :
Rules          :
accept 192.168.0.0/24
default deny
Is that ok ? (y/n) [y]
user added.
```

Каждому пользователю приданы правила, ограничивающие набор систем, которые ему можно тестировать. Два простых правила, указанных выше, позволяют Joe сканировать системы подсети 192.168.0.0/24 и ничего более.

Серверу также нужен SSL-сертификат для аутентификации у клиента. *Nessus* поставляется с готовым сертификатом, однако если вы хотите создать свой собственный, используйте команду *Nessus-mkcert*.

Теперь вы готовы к запуску демона *Nessus*. На Fedora я могу запустить демон вручную так:

```
# service nessusd start
```

Чтобы выполнить проверку, запустите графический клиент:

```
$ /usr/X11R6/bin/NessusClient
```

Для начала выберите File->Connect и подсоединитесь к серверу с ранее созданным именем пользователя. При первом подключении вам будет предложен SSL-сертификат сервера, его нужно проверить вручную и принять. Как только вы соединитесь, будут скачаны доступные модули (plugins), и вы можете выбрать, какие из них запускать. Пример показан на **рис. 2** (*слева*); правда, там нет флажков для выбора теста (они скрыты под окном, детально описывающим конкретный тест).

Зачем модули? *Nessus* использует их для проведения проверок. Это скрипты, написанные на специальном языке NASL (*Nessus* Attack Scripting Language). На моей Fedora Core модули хранятся в **/opt/Nessus/lib/nessus/plugins** (там их больше 11 тысяч!). На **рис. 2** они представлены в виде дерева. Под категорией SUSE Local Security Checks их, например, штук 200. Щелкните на строке модуля, чтобы увидеть описание соответствующего теста – это как раз и показано на рисунке. Используйте экран Выбор цели (Target Selection) для указания машины – объекта сканирования.

Есть и другие интересные вкладки: например, экран сертификатов позволяет выбрать сертификаты для SMB- и SSH-соединений. Короче, стоит потратить некоторое время для исследования этих вкладок, чтобы взять от *Nessus* все.

Настроив сканирование по вкусу, выберите Scope > Execute для его запуска. *Nessus* покажет вам индикаторы для каждой машины.

Экран отчета Nessus

Завершив сканирование, можете просмотреть экран отчета, показанный на **рис. 3** (внизу). Укажите подсеть, затем отдельный хост – увидите список открытых портов, а напротив каждого из них – иконку критичности. Выберите порт и просмотрите информацию об обнаруженных проблемах с безопасностью, их критичности и существующих способах устранения.

Nessus начинает со сканирования открытых портов на машине-объекте. Для каждого открытого порта модуль *find_service* пытается распознать, что на нем запущено, сначала пробуя SSL-соединения, затем обычные, после чего посылая различные данные сервису и распознавая ответы.

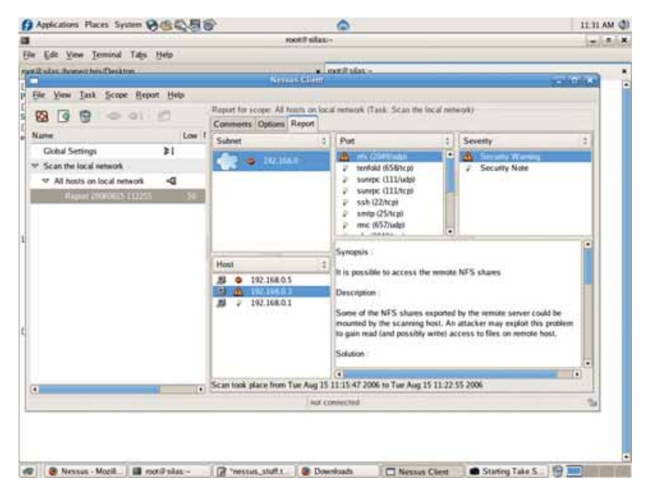
Потом *Nessus* разными способами атакует каждый открытый порт. Например, если на открытом порту найден HTTP- или HTTPS-сервер, будут запущены все модули, содержащие цель «web-сервер»: они проверят удаленные сервисы, пытаясь эксплуатировать известные уязвимости, типа неверной проверки ввода, переполнения буфера, неудачной конфигурации и так далее.

Цитирую фразу из одного шоу, которая звучит примерно так: «Знаю, который час: вчера я его записал». Моментально устаревает не только значение точного времени. Новые уязвимости в программном обеспечении находят ежедневно, и Tenable Network Security дает возможность обновить модули. Еще когда вы регистрируетесь, чтобы скачать *Nessus*, на вашу электронную почту высылается код активации. Получив его, запустите

```
# nessus-fetch --register <ваш код активации>
```

Ваш код зарегистрируется, и будут скачаны самые свежие модули. В дальнейшем нужно будет регулярно запускать **nessus-update-plugins** (например, как ежедневную задачу Cron), чтобы поддерживать программы в актуальном состоянии.

Вопреки своей мощности и популярности, *Nessus* не слишком-то документирован, но одна книга есть: *Nessus Network Auditing*, автор –



› (Рис. 3) Панели на экране отчетов *Nessus* позволят вам покопаться во всех деталях.

Бойтесь сканеров

Будьте осторожны при сканировании действующих серверов с помощью *Nessus* – некоторые типы сканирования (по умолчанию они отключены) выполняют атаки типа «отказ в обслуживании», способные вывести сервер из строя. Лучшее время выполнения подобных тестов – перед тем, как сервер будет подготовлен к реальной работе. Если вам потребуется просканировать его позже, выберите время, когда временная недоступность сервера вызовет минимальные неудобства.

Oval Test Results				
Vulnerable	Not Vulnerable	Unknown		
OVAL ID	CVE ID	Description	Status	Version
oval:org.mitre.oval:def:1001	CVE-2004-0417	Integer overflow in the "Max-dotdot" CVS protocol command (serve_max_dotdot) for CVS 1.12.x through 1.12.8, and 1.11.x through 1.11.16, may allow remote attackers to cause a server crash, which could cause temporary data to remain undeleted and consume disk space.	ACCEPTED	1
oval:org.mitre.oval:def:1003	CVE-2004-0418	serve_notify in CVS 1.12.x through 1.12.8, and 1.11.x through 1.11.16, does not properly handle empty data lines, which may allow remote attackers to perform an "out-of-bounds" write for a single byte to execute arbitrary code or modify critical program data.	ACCEPTED	1
oval:org.mitre.oval:def:1006	CVE-2004-0519	Multiple cross-site scripting (XSS) vulnerabilities in SquirrelMail 1.4.2 allow remote attackers to execute arbitrary script as other users and possibly steal authentication information via multiple attack vectors, including the mailbox parameter in compose.php.	ACCEPTED	1
oval:org.mitre.oval:def:1012	CVE-2004-0520	Cross-site scripting (XSS) vulnerability in mime.php for SquirrelMail before 1.4.3 allows remote attackers to insert arbitrary HTML and script via the content-type mail header, as demonstrated using read_body.php.	ACCEPTED	1
oval:org.mitre.oval:def:1013	CVE-2003-0984	Real time clock (RTC) routines in Linux kernel 2.4.23 and earlier do not properly initialize their structures, which could leak kernel data to user space.	ACCEPTED	1
oval:org.mitre.oval:def:1017	CVE-2004-0003	Unknown vulnerability in Linux kernel before 2.4.22 allows local users to gain	ACCEPTED	1

› (Рис. 4) *Sussen* выводит подробные результаты выполнения тестов в форме HTML-файла. Информации ужасно много, но, как и с прочими сканерами, результаты надо прочесть – для своей же пользы.

Рено Дерезон [Renaud Deraison], он же написал большую часть *Nessus* (издание Syngress Media).

Sussen: Nessus наоборот

Другая утилита для сканирования уязвимостей, на которую я недавно наткнулся – *Sussen* (да, это *Nessus* наоборот). *Sussen* запускает набор тестов уязвимостей, определенных в специальном файле, написанном на языке OVAL (Open Vulnerability and Assessment Language), и содержит три исполняемых файла: *sussen-agent* (утилита, выполняющая тесты и представляющая результаты в браузере), *Sussen-applet* (просто апплет для Gnome, запускающий *Sussen* без обращения к командной строке) и *Sussen-editor* (графический редактор OVAL-файлов). *Sussen* – из серии новомодных приложений, написанных на C#/.NET, и требует, помимо прочего, последней версии Mono и *glibc 2.4*, что делает его «крепким орешком» для большинства современных дистрибутивов. При содействии создателя *Sussen*, Лорена Бандьера [Loren Bandiera], мне удалось заставить его работать на Fedora Core 5. На **рис. 4** (вверху) показан пример вывода.

Sussen еще не завершен, особенно его редактор. Вы можете вводить новые определения и тесты, но не можете прочесть их из файла и сохранить их в файле (пока). Подробно о *Sussen* – на **http://dev.mmgsecurity.com/projects/sussen**.

Согласно сайту OVAL (**http://oval.mitre.org**), он «создает возможность взаимодействия между продуктами по безопасности, предоставляя стандартный язык на базе XML для обмена информацией». Основная идея здесь – отделить информацию об уязвимости системы от программ, в которых она используется. Обычно описание уязвимости выглядит как «Если у вас версия X программы Y, то вы подвержены уязвимости Z», однако на XML это простое определение требует не одной дюжины строк.

Фактически, *Sussen* – просто движок для запуска тестов уязвимостей, определенных в OVAL-файлах. Могу сказать, что он прилично выполняет эту работу, однако эффективность утилиты зависит в первую очередь от самих OVAL-файлов. В Red Hat, похоже, приняли идею; они публикуют обширные отчеты, используя OVAL (**www.redhat.com/oval**), хотя непохоже, чтобы в SUSE (например) делали то же самое. Я считаю OVAL хорошей идеей, но не уверен, что она привлечет критическую массу пользователей для повсеместной реализации.

Есть и другие сканеры уязвимостей, например, Sara (*Security Auditor's Research Assistant*) и Saint (*Security Administrator's Integrated Network Tool*). Кому нужен полный список, посетите страничку Fyodor'a (автора *Nmap*) – **http://sectools.org/vuln-scanners.html** LXF

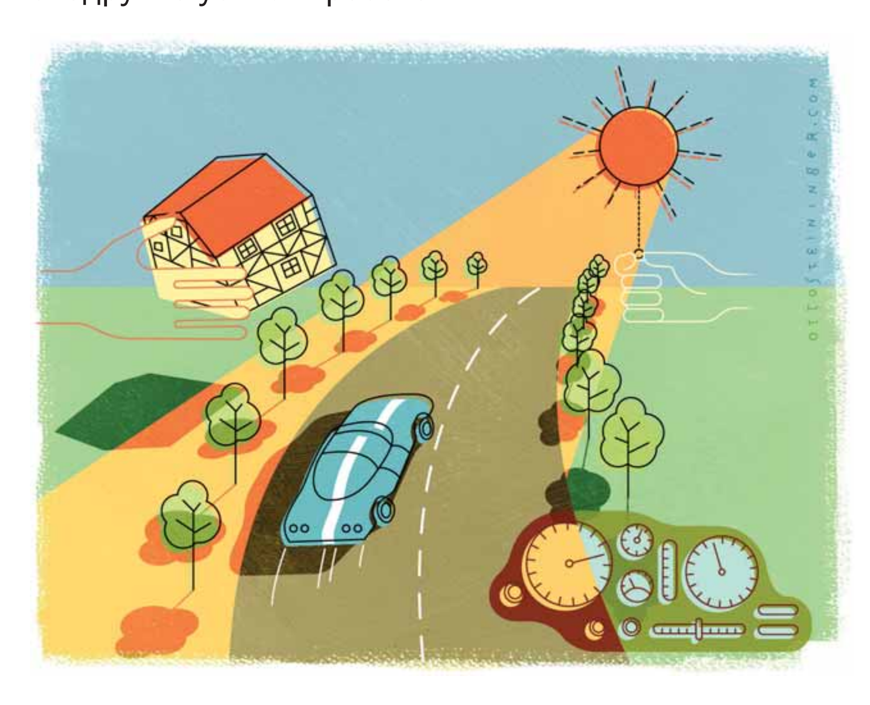


Не сканируйте системы на уязвимость у себя на работе без разрешения вашего начальника!

» Через месяц Мы создадим брандмауэр – всего на четырех страницах. Подключайтесь!

Ogre: Саундтрек

ЧАСТЬ 5: Последний урок в данной серии – музыка для ушей **Пола Хадсона**: под такую сподручно убивать роботов.





Наш эксперт

Пол Хадсон написал три книги по Linux и одну по PHP, он также поддерживает на SourceForge два проекта на Mono по лицензии GPL. Пол любит Emacs.

На данном этапе наша игра содержит все базовые элементы стрелялки от первого лица, но имеет легкий недостаток: подстрелить-то вы никого и не можете. Пожалуй, это скорее тяжелый недостаток, если учесть, что *Висельник Чед* для стрельбы и задуман. Не хватает также звука и музыки, да и прицела оружия, чтоб видеть, куда мы стреляем. А фанаты C++, наверно, заметили, что отсутствует какое-либо высвобождение памяти.

Изящно завершим *Висельника Чеда*: реализуем все эти элементы на следующих четырех страницах, и притом запросто – обещаю!

Включаем громкость

Имя *Ogre* не дает забыть о сильных сторонах программы: это акроним, означающий Объектно-ориентированный Графический Движок Рендеринга [Object-Oriented Graphics Rendering Engine]. Звук – как для эффектов, так и фоновый – не принимается в расчет и, согласно разработчикам *Ogre*, приниматься не будет. Но это не проблема, благодаря библиотеке *SDL* и ее расширению *SDL_Mixer*: вместе они позаботились о поддержке аудио. Если вы следили за нашими уроками с LXF82, то уже установили библиотеки *libsdl-devel* и *libsdl-mixer-devel*; а те, кто этот номер пропустил, пусть начнут с их установки, иначе код данного урока работать не будет.

Прежде всего, надо изменить файл **chad.h**, объявив в нем звуковые файлы стрельбы (я использую **laser1.wav**) и музыкального фона (**tipperary.mp3**). В *SDL*-терминах это Mix_Chunk и Mix_Music соответственно, поэтому добавьте две строки в конец класса CChadGame:

```
Mix_Chunk* m_MixFire;
Mix_Music* m_Music;
```

Загрузка нашего аудиоматериала осуществляется в файле **chad.cpp**, в методе initialise(). В конец этого метода (т.е. после установки m_SceneMgr в NULL), добавьте следующие четыре строки:

```
SDL_Init(SDL_INIT_AUDIO);
Mix_OpenAudio(44100, AUDIO_S16SYS, 2, 2048);
m_MixFire = Mix_LoadWAV("laser1.wav");
m_Music = Mix_LoadMUS("tipperary.mp3");
```

Первая строка инициирует поддержку звука, потому она и идет первой. Функция SDL_Init() сообщает *SDL*, какие части вы хотите использовать – графику, звук, ввод, таймеры и т.д., обычно через передачу списка констант, объединенных оператором ИЛИ – например, SDL_INIT_AUDIO | SDL_INIT_TIMER | SDL_INIT_CDROM. Для инициализации всех доступных в библиотеке подсистем (это изрядная расточительность, если вы не намерены все их использовать!), просто укажите SDL_INIT_EVERYTHING.

Инициализировав звуковую подсистему *SDL*, можно открывать звуковое устройство. Это делает функция Mix_OpenAudio(): у нее четыре параметра, определяющих свойства звука. Первый параметр – частота дискретизации: 44100 соответствует CD-качеству; чтобы игра лучше работала на старых компьютерах, попробуйте уменьшить ее до 22050. Второй параметр – формат сэмпла (AUDIO_S16SYS означает 16 бит, какой байт старший – определяется системой), третий – количество каналов (1 для моно, 2 для стерео), а четвертый определяет размер буфера для проигрывания звука. Вам эти параметры ни о чем не говорят? Можете их проигнорировать. Просто скопируйте и вставьте приведенную мной строку кода и больше о ней не вспоминайте. Магия!

Запускаем звуковые файлы

Настроив звуковую систему, мы, наконец, можем заказывать наши звук и музыку. *SDL*_Mixer берет на себя их загрузку, а вам остается сделать два вызова функций Mix_LoadWAV() и Mix_LoadMSU(). Они принимают имя загружаемого файла и автоматически обрабатывают множество популярных форматов – WAV, MP3, OGG, MID и MOD, но если ваш дистрибутив не поддерживает формат MP3, то *SDL*, скорее всего, не сможет его проиграть. Кому интересно, общедоступную запись **tipperary.mp3** я нашел в Сети – она совершенно не подходит для игры [«Путь далекий до Типперери» – популярная песенка английских солдат времен I Мировой войны, – прим. ред.], поэтому вы уж сами подберите нужный файл!

Чтобы покончить с поддержкой звука, остается еще два шага. Добавьте в методе frameStarted() следующие три строки кода:

```
if (!Mix_PlayingMusic()) {
		Mix_PlayMusic(m_Music, 0);
}
```

Я не собираюсь вас унижать, объясняя этот код, кроме 0 в конце: это число повторов нашей мелодии [0 значит, что она будет проиграна всего один раз, без повтора, – прим. ред.].

Добавление звука лазера потребует немного мозгов, поскольку потребуется определить метод mousePressed(). В настоящий момент он пуст и сидит в **chad.h**. Заменим «заглушку» в **chad.h** на прототип и

Месяц назад Мы познакомились с кватернионами, ИИ и Стандартной Библиотекой Шаблонов.

для лазера

напишем реализацию этого метода в **chad.cpp** (чтобы лазер зазвучал). В **chad.h**, превратим строку....

```
void mousePressed(MouseEvent* e) { }
```

... в...

```
void mousePressed(MouseEvent* e);
```

Тело этого метода надо поместить где-то в файле **chad.cpp**:

```
void CChadGame::mousePressed(MouseEvent* e) {
        Mix_PlayChannel(-1, m_MixFire, 0);
}
```

Звук лазера теперь будет раздаваться при каждом нажатии кнопки мыши – не сногсшибательно, но начало хорошее! Можете скомпилировать свой код и насладиться звуками лазера.

Прицел

Если вы не снайпер сразу после дембеля, то вряд ли поражение цели на дальней дистанции покажется вам несложным. Для упрощения этой задачи многие игры содержат на экране небольшой прицел. Добавим и мы прицел в виде точки в игру *Висельник Чед*. Для этого необходимо проделать три шага:

1. Создать материал, который даст имя файлу.
2. Создать слой, который использует материал, и позиционировать его на экране.
3. Отобразить слой.

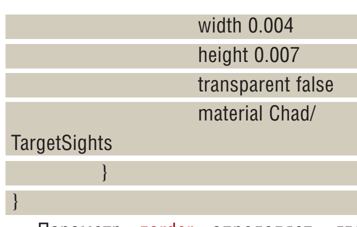
Первые два пункта реализуются через систему скриптов *Ogre*; но для последнего шага придется написать код на C++. Начнем с материала. Сохраните следующий 'код' как **target.material**:

```
material Chad/TargetSights
{
        technique
        {
                pass
                {
                        lighting off
                        scene_blend alpha_blend
                        texture_unit
                        {
                                texture terrain_detail.jpg
                        }
                }
        }
}
```

Заметили? Я использовал для прицела текстуру **terrain_detail.jpg**, но только потому, что прицел очень мал: игроки увидят лишь небольшую серую точку. Вы можете взять свою картинку, но пока сойдет и эта.

Следующий шаг – определить слой, который принимает материал и помещает его на экранной панели. Затем можно пристроить эту панель на экране, используя координату относительно левого верхнего угла, а также ширину и высоту. В любом случае, вот код – сохраните его в файл **target.overlay**:

```
chadtarget
{
        zorder 650
        container Panel(chadsight)
        {
                metrics_mode relative
                left 0.495
                top 0.495
                width 0.004
                height 0.007
                transparent false
                material Chad/
TargetSights
        }
}
```



Параметр zorder определяет, где панель отобразится на экране, в терминах глубины, то есть мы можем (если пожелаем) задать способ расположения элементов по слоям. В *Ogre* его максимальное значение *650* (прицел на вершине стека).

Последний шаг – создание слоя, он займет всего две строки. Добавьте такой код в конец метода createOutdoorScene():

```
Overlay *TargetSight = (Overlay*)OverlayManager::getSingleton().
getByName("chadtarget");
TargetSight->show();
```

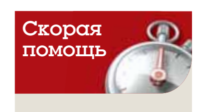
Код загружает слой с именем, определенным в **target.overlay**, а затем отображает его. Не надо беспокоиться о потере указателя на слой – *Ogre* автоматически удерживает его посреди экрана, как определено в файле.

Наши три шага проделаны; запустите игру и загляните в прицел. Пусть программировать было скучновато, но зато как удобно теперь целиться!

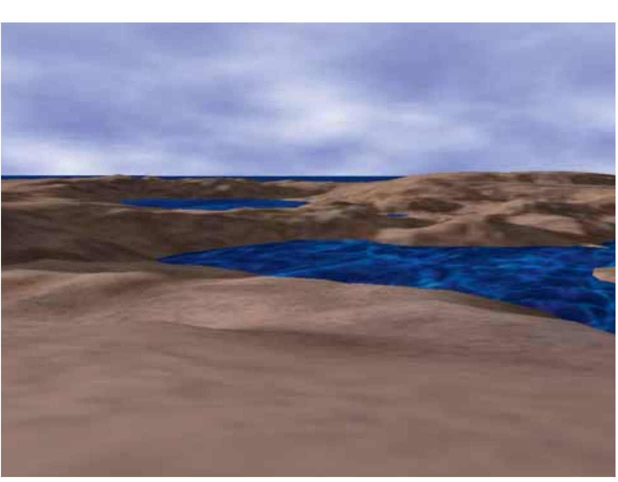
Стреляем на поражение

Настает главное событие этого урока: отстрел роботов, которые резвились в прошлом номере. Правду сказать, я не особо хотел отягчать насилием *Висельника Чеда*, но Ребекка – спец по насилию в нашей команде – отказалась плодить опечатки, пока мы не разнесем когонибудь на куски. Пусть будет так. У нас уже есть метод mousePressed() для проигрывания звука лазера, поэтому код для стрельбы подойдет именно сюда.

Стрелять будем так: с позиции камеры проводим луч, аналогично тому, как мы делали для определения высоты игрока над уровнем земли. Это непростая геометрическая задача, но, к счастью, *Ogre* все делает »

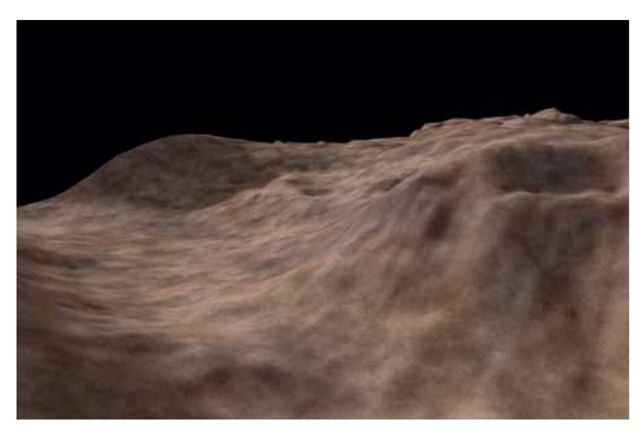
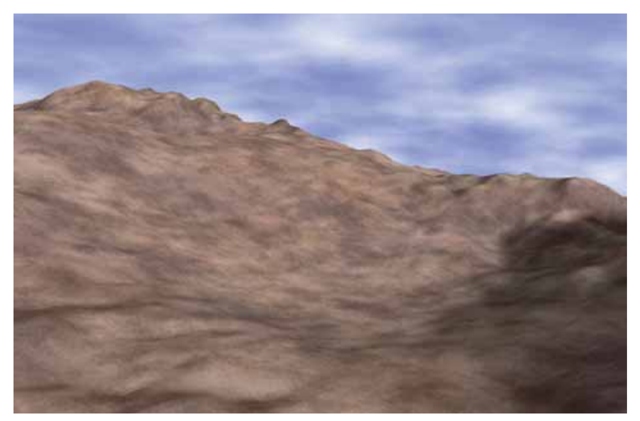


Мы сдержали свое обещание сделать наш код кросс-платформенным: можете скомпилировать игру и играть в нее на любой платформе – включая Windows!



› Не забывайте о методе создания волн, посредством изменения высоты водной поверхности каждые несколько кадров.



› Далеко мы забрались: взяв готовые текстуры Ogre, создали ландшафт, а затем добавили эффект смены дня и ночи.

» сам, одним методом: getCameraToViewportRay(), который преобразует позицию на экране в позицию в нашем мире и позволяет пустить луч из положения камеры. При необходимости выбирать объекты мышью, можно было бы использовать указатель на объект-событие, передаваемый методу mousePressed(), но мы хотим просто пустить луч через центр экрана, и поэтому используем для координат X и Y значения *0.5*.

Вот код улучшенного метода mousePressed():

```
void CChadGame::mousePressed(MouseEvent* e) {
        Mix_PlayChannel(-1, m_MixFire, 0);
        Ray mouseray = m_Camera->getCameraToViewportRay(0.5,
 0.5);
        RaySceneQuery* scenequery = m_SceneMgr
->createRayQuery(Ray());
        scenequery->setRay(mouseray);
        RaySceneQueryResult &result = scenequery->execute();
        if (!result.empty()) {
            for (unsigned int i = 0; i < result.size(); ++i) {
            RaySceneQueryResultEntry &re = result[i];
            if (re.movable && re.movable->getMovableType() ==
"Entity") {
                Entity *ent = (Entity*)(re.movable);
                String name = ent->getName();
                if (name == "water") continue; // игнорируем воду
                for (unsigned int j = 0; j < Enemies.size(); ++j) {
                    if (Enemies[j]->m_EnemyName ==
 name) {
                        Enemies[j]->Hit();
                        return;
                        }
                    }
                }
            }
        }
}
```

Послав луч, я прошелся в цикле по откликам в поисках сущностей (в отличие от элементов ландшафта), а затем отсек водную поверхность. Остаются только роботы; получим имя жертвы и найдем в списке роботов соответствующий объект, а затем вызовем его метод Hit().

Те, кто следил за нашими уроками с самого начала, возможно, воскликнет: у наших врагов нет ни имени, ни метода Hit()! Исправим это: откройте файл **chadenemy.h** и добавьте эти строки до объявления m_Speed:

```
char m_EnemyName[32];
bool m_IsDead
```

Теперь добавьте следующие две строчки после метода Update():

```
void SetAnimation(String animation, bool loop);
void Hit();
```

Метод SetAnimation() я вставил, потому что мне было тошно писать три строчки кода для выполнения одной простой вещи. Можете игнорировать его, если хотите.



Мы уже устанавливали имена врагов в конструкторе (файл **chadenemy.cpp**), но использовали временную локальную переменную. Теперь, когда у нас есть m_EnemyName, мы можем хранить ее там, чтобы найти правильный объект сцены, попав в робота. Просто замените следующие три строки кода…

```
char enemyname[32];
sprintf(enemyname, "Robot %d", ++EnemyNum);
m_Entity = m_SceneMgr->createEntity(enemyname, "robot.
mesh");
```

… на следующие две…

```
sprintf(m_EnemyName, "Robot %d", ++EnemyNum);
m_Entity = m_SceneMgr->createEntity(m_EnemyName, "robot.
mesh");
```

Теперь нужно написать метод Hit(), изменить анимацию робота и установить переменную m_IsDead в значение true, например, так:

```
void CChadEnemy::Hit() {
        if (!m_IsDead) {
                m_AnimationState->setEnabled(false);
                SetAnimation("Die", false);
                m_IsDead = true;
        }
}
```

То есть, если чудовище не сдохло на месте, отмените его текущую анимацию, измените его анимацию на «умирающий» (не в цикле, конечно), а затем добейте его.

Теперь необходимо сказать игре, что делать, если робот убит. Для начала сделаем, чтобы убитые роботы переставали двигаться, освободив метод Update(), раз уж робот встретился со своим создателем. Далее, в методе Update() необходимо удалить робота-покойника из игры. Иначе все убитые роботы будут валяться вокруг – но, возможно, вам того и надо! В файле **chadenemy.cpp** измените начало метода Update() на следующее:

```
void CChadEnemy::Update(Real time) {
        m_AnimationState->addTime(time);
        if (m_IsDead) return;
```

Здесь все понятно: мы хотим, чтобы робот продолжал проигрывать анимацию (чтобы анимация «Умер» отработала правильно), но не хотим, чтобы он перемещался или поворачивался к игроку. Если робот внезапно умер, мы скоренько выходим из функции.

Обломки роботов

Удалить мертвых роботов из вектора Enemies желательно элегантным способом, так что потребуется немного подумать. Если робот погиб, и текущее положение в анимации равно ее полной длине, освободим память и присвоим его сущности в массиве Enemies значение *NULL*. Затем используем метод erase() из библиотеки STL, чтобы удалить из вектора всех обнуленных врагов, а это, в свою очередь, требует использования особого алгоритма STL – remove_if. Звучит хитроумно, но все именно так: STL умеет автоматически удалять элементы из вектора на основе заданного критерия. Можно написать функцию, при-

нимающую элемент вектора, и если функция возвратит значение *true*, элемент будет удален.

Цикл обновления состояния врагов в методе frameStartedOutside() файла **chad.cpp** нужно привести к следующему виду:

```
for (unsigned int i = Enemies.size() - 1; i > 0; --i) {
        CChadEnemy* enemy = Enemies[i];
        enemy->Update(evt.timeSinceLastFrame);
        if (enemy->m_IsDead && enemy->m_AnimationState
->getTimePosition() == enemy->m_AnimationState->getLength()) {
                m_SceneMgr->getRootSceneNode()-
>removeAndDestroyChil
d(enemy->m_EnemyName);
                delete Enemies[i];
                Enemies[i] = NULL;
        }
}
Enemies.erase(remove_if(Enemies.begin(), Enemies.end(), IsNull),
Enemies.end());
```

Мы перешли от использования цикла с итератором к циклу с целочисленным счетчиком, чтобы манипулировать отдельными элементами вектора Enemies. Убедившись, что враг мертв и доиграл свою анимацию до конца, мы вызываем метод removeAndDestroyChild() корневого узла сцены и выносим из нее покойника. Затем роботу присваивается значение *NULL* и освобождается занимаемая им память. После цикла вызываются чародеи erase() и remove_if(); последний принимает как параметры начало и конец диапазона, а также функцию обратного вызова. Эта строка пройдется по всему вектору Enemies и удалит элементы, которым при передаче их в функцию IsNull() возвращается значение *true*.

Недостает только самой функции IsNull(), но с ней разобраться проще всего. Откройте файл **chad.h** и добавьте следующую строку к списку директив #include:

```
#include <algorithm>
```

Эта строка задействует алгоритм remove_if. Теперь добавьте такую строку сразу после 'using namespace *Ogre*':

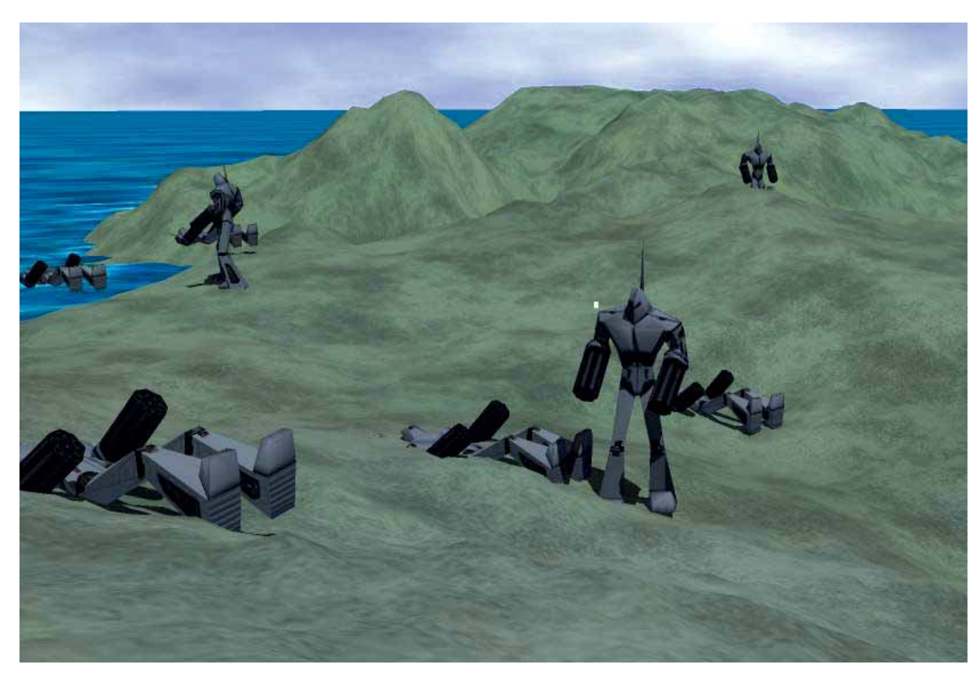
```
bool IsNull(void* somepointer) { return (somepointer == NULL); }
```

Она отвечает на вопрос: не *NULL* ли значение некого указателя? Просто и мило.

Вместо заключения

На протяжении пяти уроков мы прошли путь создания 3D-игры с музыкой, звуками и врагами. Я думаю, неплохо. Причем согласитесь, вопросы математики мы почти не обсуждали. Может, от кватернионов у вас голова и пошла кругом, но в основном за вас думал *Ogre*.

Надеюсь, вы кое-что узнали насчет процесса создания 3D игры – в частности, как просто, немного потрудившись, получить солидные результаты. И действительно, написав 500 строк кода, мы уже можем чем-то гордиться: пусть это не образцовая игра, но хороший демо-пример и отличная отправная точка для разработки таковой. Может, вам захочется создать новые уровни, добавить врагов или изменить цель игры в Чеда. Желаем удачи – потом расскажете, как далеко вы продвинулись! LXF



› **Браво… сцена завалена мертвыми и умирающими роботами, а Чед выходит победителем.**



Благодарности



Видеокарта Nvidia GeForce 7800, использованная при разработке этого урока, была любезно предоставлена фирмой MSI – спасибо, ребята!

Очистка памяти

Я признаю, что люблю что-то делать, но не люблю убирать за собой. Возможно, именно поэтому мне нравится программировать под Mono – он автоматически подбирает за вами весь мусор! Пока что мы не беспокоились об освобождении отведенной памяти, поскольку Linux освобождает ОЗУ автоматически. Но если вы планируете что-то динамически подгружать во время игры, тут уж об освобождении памяти придется позаботиться. Если вы, например, запустите звук и музыку, то в конце концов память будет исчерпана.

Ogre прекрасно выполняет свои задачи, а вот памятью управляет плоховато, ибо пытается вмешаться в управление. Под C++ освобождение памяти не слишком поддается интуиции: вы беспокоитесь об одном, а нужно вовсе другое. Не тратьте время на попытки понять, что там вытворяет *Ogre*, лучше займитесь собственными обязанностями.

Они сводятся к выполнению правила: при каждом применении new для создания нового объекта, не забывайте использовать delete для его удаления. В методе run() у нас есть следующая строка:

```
m_Player = new CChadPlayer();
```

В методе CChadGame::~CChadGame() необходимо добавить следующее:

```
delete m_Player;
```

SDL имеет собственные функции для освобождения аудиоресурсов, и понадобится вставить туда же такие две строчки:

```
Mix_FreeChunk(m_MixFire);
Mix_FreeChunk(m_Music);
```

Kamaelia: P2P

Майкл Спаркс расскажет, как запустить децентрализованное whiteboard-приложение, используя его новый каркас, а затем расширить его до мультимедиа-вещания и так далее.



Kamaelia – это открытый каркас общего назначения для разработки программ. Вы скажете: ну вот, еще один… Но Kamaelia имеет отличие: она работает с распределенными сетями в стиле *BitTorrent* для поддержки общего доступа к информации в децентрализованной [peer-to-peer] сети в режиме реального времени. Приложения, разрабатываемые с помощью Kamaelia, умеют мгновенно доставлять информацию, через LAN или интернет, потенциально неограниченному числу машин.

Эта технология возникла благодаря исследованиям BBC в сфере сетевой передачи мультимедийного контента, но стала применяться и в других областях. Она включает компоненты для работы с Freeview (свободное цифровое наземное телевидение в Великобритании) и инструменты обработки мультимедиа, и позволяет разработчикам легко и просто использовать Pygame, OpenGL, сетевые возможности, видеокодек Dirac, Vorbis, Speex и множество других инструментов в одной и той же системе.

Данный урок покажет, как пользоваться whiteboard-приложением, написанным для решения реальных проблем команды Kamaelia. [Whiteboard – приложение, позволяющее пользователям, объединенным сетью, рисовать на одной «доске». В русском языке встречаются термины «белая доска», «разделяемый блокнот», «разделяемая калька». Мы будем говорить «блокнот», – прим. перев.] Наша команда разбросана по разным местам и нуждается в системе совместной работы, поддерживающей и звук, и эскизы, и просто работы – а это может пригодиться многим открытым проектам. Некоторые из возможностей системы описаны во врезке «Что предлагает наш блокнот», справа. Интересно? Тогда займемся.

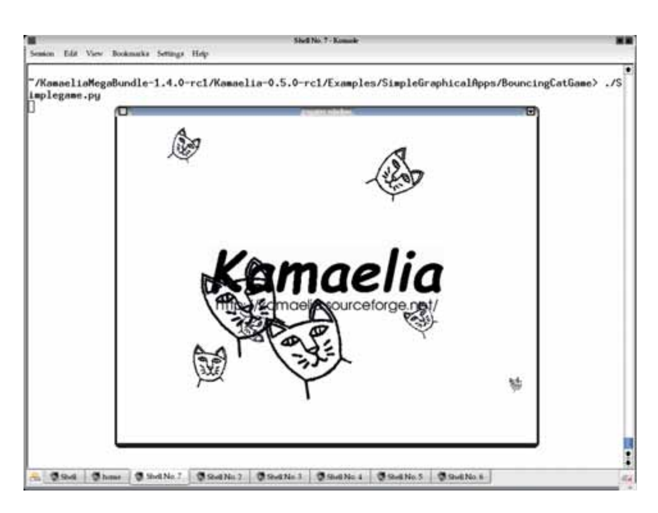


Наш эксперт

Майкл Спаркс ведущий инженер BBC Research и лидер открытого проекта *Kamaelia*. Данный урок и интервью отражают его личное мнение как лидера проекта *Kamaelia*, а не мнение BBC относительно чего бы то ни было.

Часть 1. Установка ПО

Kamealia разработана для SUSE Linux, с использованием Python 2.4. Она должна работать под любым дистрибутивом Linux; но вашу конфигурацию мы вряд ли проверяли – обратная связь с командой разработчиков приветствуется (http://kamaelia.sourceforge.net/Contact.html). Kamaelia должна работать с любой версией Python, начиная с 2.2.2, но мы рекомендуем 2.4. Для установки Kamaelia требуются два основных пакета: ядро системы, называемое **Axon**; и библиотека компонентов и инструментов **Kamaelia**. После них устанавливайте любые дополнительные зависимости для желаемых специфических функций (примеры см. во врезке «Зависимости Kamaelia», *справа*). Для удобства я предполагаю, что вы выполняете установку как root. Кстати, вы можете считать **Axon** аналогом ядра Linux, а **Kamaelia** – базовой инсталляцией; все остальные файлы – необязательные дополнения.



› **В Kamaelia легко делать простые приложения, типа этого «демо скачущих котов». Логотип Kamaelia написан от руки.**

Добавляем ингредиенты, по очереди

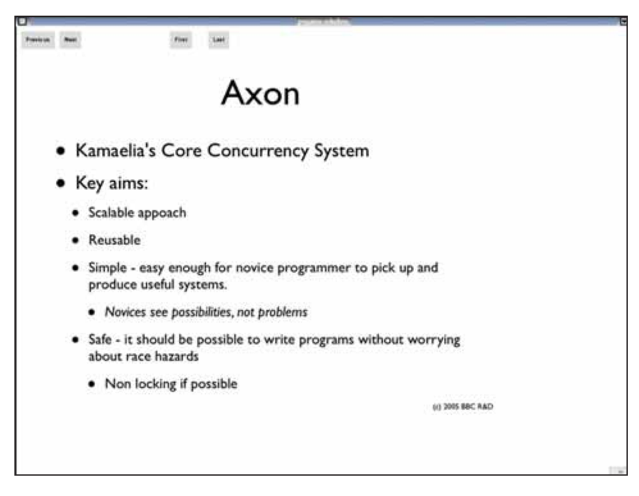
» **Axon**. Сначала установите **Axon**, чтобы обеспечить базовый коммуникационный каркас для компонентов. Распакуйте, перейдите в каталог и запустите инсталлятор:

```
tar zxf Axon-1.5.1.tar.gz
cd Axon-1.5.1
python setup.py install
```

» **Kamaelia**. Затем установим репозиторий компонентов. Распакуйте еще один архив, перейдите в каталог и запустите инсталлятор:

Месяц назад: Мы копались в крутых наворотах рабочего стола на базе *Compiz*.

сотрудничество



› Граф-канал» для слайд-шоу в действии (см. кнопки слева вверху). О его сборке см. стр. 65

Что предлагает наш «блокнот»

» Простейшие функции рисования.
» Поддержка многостраничности с навигацией.
» Каждый «блокнот» может быть клиентом, сервером или сразу и клиентом, и сервером. Если вы подключите свой «блокнот» к моему, а я изменю страницы, вы увидите эти изменения. Все, нарисованное в одном «блокноте», становится видимым на всех подключенных к нему. Поскольку любой «блокнот» может быть и сервером, и клиентом, он является децентрализованным в своей основе. Каждый подключившийся к любому серверу в сети может получить все, что предоставляется остальными.
» Поддержка звука с использованием аудиокодека Speex (оптимизированного для разговора и для кодирования с очень низкой частотой дискретизации). То есть можно поговорить с любым из подключенных в данный момент.
» Сессии могут быть записаны для последующего воспроизведения. Пока запись воспроизводится, вы можете использовать «блокнот» как обычно – как и все подключенные к вашей whiteboard-сети.
» Поддержка MP3.
» Консоль с командной строкой.
» Экспериментальная возможность заставить «блокнот» нумеровать локально сохраненные страницы полусинхронным образом.

```
tar zxf Kamaelia-0.5.0.tar.gz
cd Kamaelia-0.5.0
python setup.py install
```

» **Pygame**. Блокнот использует Pygame для отображения и ввода. Pygame может быть уже установлен у вас или включен в ваш дистрибутив. Если его нет, или ваша версия не последняя, установка выполняется легко: распакуйте, перейдите в каталог и запустите инсталлятор:

```
tar zxf pygame-1.7.1release.tar.gz
cd pygame-1.7.1release
python setup.py install
```

Вам будет задано несколько вопросов, зависящих от вашей локальной установки, вот почему предпочтительней использовать пакеты из вашего дистрибутива!

Поддержка мультимедиа

» **Speex**. «Блокнот» использует аудиокодек Speex для эффективной передачи звука. Сперва установите Speex, затем – PySpeex, а затем – PyMedia. (Speex используется непосредственно для кодирования, PySpeex для активации компонентов Speex, а PyMedia – для ввода/вывода). Установка Speex вполне стандартна – распакуйте **speex-1.0.5.tar.gz**, перейдите в полученный каталог, введите **./configure**, затем – **make** и **make install**. Теперь нужно установить поддержку Speex для Python, распаковав **PySpeex-0.2.tar.gz**, перейдя в него и выполнив **python setup.py install**. Распакуйте предоставляемую версию PyMedia с наложенными заплатками и установите, как было описано выше.

» **Python Image Library**. Устанавливается так же, как **Axon** и **Kamaelia**: распаковать **Imaging-1.1.5.tar.gz**, перейти в полученный каталог и выполнить **python setup.py install**.

В этом уроке я остановился только на зависимостях для «блокнота»; но в MegaBundle полно других. Для более подробных инструкций по всем зависимостям загляните в отдельные пакеты или зайдите на http://kamaelia.sourceforge.net/GettingStarted.html. В Kamaelia существуют PVR-инструменты для захвата ТВ-сигнала и временного разделения.

Если все прошло успешно, в вашей системе будет установленная Kamaelia. Если натолкнетесь на проблемы, проверьте, предоставляет ли ваш менеджер пакетов соответствующие версии ПО.

Зависимости Kamaelia

Kamalelia MegaBundle (http://snipurl.com/xie4) включает все основные зависимости. В ее недрах можно найти:

» Обязательные файлы: **Axon-1.5.1.tar.gz, Kamaelia-0.5.0.tar.gz**.
» Рекомендуемые файлы: **pygame-1.7.1release.tar.gz, Pyrex-0.9.3.1.tar.gz**.
» Для поддержки звука в «блокноте»: **speex-1.0.5.tar.gz, pySpeex-0.2.tar.gz, pymedia-cvs-patched.tar.gz**.
» Для загрузки и сохранения изображений: **Imaging-1.1.5.tar.gz**.
» Для *BitTorrent*: **BitTorrent-4.20.8.tar.gz**.
» Для *OpenGL*: **PyOpenGL-2.0.2.01.tar.gz**.
» Поддержка Dirac: **dirac-0.5.4.tar.gz**.
» Поддержка Vorbis (устанавливается в указанном порядке, в дополнение к Pyrex): **libao-0.8.6.tar.gz, libogg-1.3.3.tar.gz, libvorbis-1.1.2.tar.gz, pyao-0.82.tar.gz, vorbissimple-0.0.2.tar.gz**.
» Freeview (вам понадобится Pyrex и последнее ядро Linux): **python-dvb3-0.0.4.tar.gz**.

Набор зависимостей довольно велик, но следует помнить, что они требуются только для поддержки дополнительных функций. Не нужен Dirac – не устанавливайте зависимости Dirac!

Часть 2. Использование «блокнота»

«Блокнот» предусматривает запуск из каталога установки. Однако можно переместить каталог «блокнота» в любое удобное для вас место вашей системы. Запустите приложение командами:

```
cd Kamaelia-0.5.0-rc1/Tools/Whiteboard/
./Whiteboard.py
```

Если все завершилось нормально, вас встретит чистый экран и небольшая палитра цветов для рисования, а также ластик. Система будет захватывать звук, но поскольку мы запустили «блокнот» в автономном режиме (т.е. не в сети), он никуда не будет передаваться.

Для начала, набросайте что-нибудь на первой странице. Для создания новых страниц, щелкайте на new page. Для перемещения назад и вперед между страницами, щелкайте на << и >>. Все, что вы пишете »

Скорая помощь

Kamaelia спроектирована для упрощения поддержки параллельных программ (это реальная область для исследований!). Старые Unix-хакеры согласятся, что «граф-каналы» [graphline], подобно обычным [pipeline], заведомо способны стать полностью параллельными.

» на страницах, сохраняется, т.е. можно вернуться назад и увидеть это, даже если вы поменяете страницы. Эта функция означает, что использование «блокнота», особенно на ноутбуках-планшетах или внешних планшетках (т.е. таких, которые вы можете прикупить в супермаркете!), совершенно интуитивно.

Чтобы запустить whiteboard-приложение как сервер, добавьте номер обслуживаемого порта:

```
./Whiteboard.py --serveport=1500
```

Сервер теперь будет способен принимать соединения от стольких клиентов, сколько потянет ваше оборудование. Чтобы подключиться как клиент, запустите

```
./Whiteboard.py --connectto=192.168.2.5:1500
```

где **192.168.2.5** – IP-адрес сервера, а **1500** – номер порта.

Если вы выполняете тестирование на локальной машине (localhost, он же – 127.0.0.1), имеет смысл сделать копию каталога «блокнота» перед запуском второго экземпляра:

```
cd Kamaelia-0.5/Tools
cp -R Whiteboard ClientWhiteboard
cd ClientWhiteboard
./Whiteboard.py --connectto=127.0.0.1:1500
```

(Вы, понятно, можете создать копию каталога где захотите.) Смысл заключается в том, что при перемещении между отредактированными страницами изменения сохраняются, и два постоянно затирающих друг друга «блокнота» могут буквально взбесить!

Теперь, имея два «блокнота», вы можете видеть, что все нарисованное на первом дублируется на втором. Если кто-нибудь создает новые страницы локально, эти страницы видимы только ему – у каждого есть свой собственный набор локальных страниц. Коллекция страниц не синхронизируется просто потому, что мы решили, что так удобнее: тогда не нужно проверять, что вы начинаете с одного и того же набора страниц перед соединением двух «блокнотов». Вы просто подключаетесь и начинаете взаимодействовать.

С подключившимися друзьями можно еще и разговаривать. Качество звука сильно зависит от аудио-оборудования вашей машины и используемых микрофонов, так что стоит потратиться на внешний микрофон или гарнитуру. USB-микрофоны могут быть здесь особенно хороши. Если вы используете внешний микрофон, установите его как источник захвата по умолчанию в аудио-микшере вашей системы.

Клиент-сервер

Запуск «блокнота» как клиента и сервера – просто комбинация вышеприведенных опций:

```
./Whiteboard.py --serveport=1500 --connectto=192.168.2.5:1500
```

Тут «блокнот» становится настоящим узлом в терминах децентрализованной конфигурации. Однако понятия ячеистой [mesh] или древовидной [tree] конфигурации отсутствуют. Интересно было бы автоматизировать подключение к whiteboard-сети...

Возникает очевидный вопрос: «Изображения сохраняются автоматически… а где?». А внутри подкаталога **Scribbles**, как стандартные PNG-файлы (поскольку PNG, благодаря сжатию без потерь, хорошо работает с изображениями, типичными для «блокнота»). На моей системе они находятся здесь:

```
cd Kamaelia-0.5.0/Tools/Whiteboard/
ls Scribbles
slide.1.png slide.2.png slide.3.png
```

Часть 3. Запись ваших сессий

Если вы прошли предыдущие шаги, то теперь можете использовать «блокнот» для совместной работы с друзьями и коллегами – вплоть до игры в крестики-нолики. Записанная партия этой игры, возможно, не выглядит волнующе; однако фиксация страниц из вашего плана мирового господства и всей сессии его подготовки может и пригодиться. Так что рассмотрим, как записать сессию; как воспроизвести сессию; как загрузить и сохранить страницы из произвольного места на диске; и, наконец, как добавить MP3 к работающей сессии (скажем, для транскрипции).

Прежде всего, запустите whiteboard-сервер, набрав **./Whiteboard.py --serveport=1500**. Затем можно запустить программу записи, подключив ее к этому серверу. Если сервер работает на локальной машине, наберите **./WhiteboardRecorder.py whiteboard_session.rec 127.0.0.1 1500**. Если whiteboard-сервер удаленный – в нашем случае, работающий на 192.168.2.5 – надо набрать **./WhiteboardRecorder.py whiteboard_session.rec 192.168.2.5 1500**, и т.д. Для остановки записи просто нажмите Ctrl-C.

Чтобы воспроизвести запись сессии, запустите сервер, набрав **./Whiteboard.py --serveport=1500**. Затем, опять-таки, для случая с локальной машиной, примерно так же запустите инструмент воспроизведения: **./WhiteboardPlayer.py whiteboard_session.rec 127.0.0.1 1500** либо, для удаленной машины, **./WhiteboardPlayer.py whiteboard_session.rec 192.168.2.5 1500**.

Замечательно, что во время воспроизведения сессии вы можете разговаривать и малевать в «блокноте», благодаря тому, что с точки зрения системы, плейер – просто другой пользователь, подключившийся к «блокноту». Аналогично, программа записи так же получает данные, действуя как еще один клиент.

Помимо метафоры переворота страниц, система также позволяет загружать и сохранять их вручную. Если вы вернетесь к консоли, с которой запустили «блокнот», вы найдете там скромненькое приглашение командной строки. Предположим, мне понравился наш план загрузки саней, и я решил сделать себе копию записи. Я мог бы сделать это, набрав одну из следующих команд:

```
>>> SAVE '/home/michaels/plans.png'
>>> SAVE '/home/michaels/plans.jpg'
```

Аналогично, если сохраненное изображение уже есть, я могу загрузить его, заменив команду **SAVE** на **LOAD**:

```
>>> LOAD '/home/michaels/plans.png'
>>> LOAD '/home/michaels/plans.jpg'
```

Если вы хотите воспроизводить MP3 во время сессии, вам понадобится запущенный whiteboard-сервер (скажем, 192.168.2.5 на порту 1500). Затем вы вводите:

```
./MP3Player some_podcast.mp3 192.168.2.5 1500
```

Как и другие программы, MP3Player – это просто специализированный клиент, и люди могут обсуждать то, что он «говорит».



› Что происходит, когда несколько «блокнотов» объединяются, формируя сеть.

Часть 4. Научная основа Kamaelia

Kamaelia работает по принципу логического развития Unix-каналов [pipeline] на шаг вперед. Отличия заключаются в том, что вместо прямых каналов вы можете создать произвольные фигуры (которые мы называем «граф-каналами» [graphline]). Любой объект Python можно послать по ребрам этого графа – в отличие от однонаправленной файлоподобной схемы передачи данных. Кроме того, наши компоненты используют хитрость Python, позволяющую системе оставаться однопоточной. При желании, можно использовать и многопоточную архитектуру, но, в отличие от каналов Unix, нам не требуются тяжеловесные процессы. Это, естественно, в итоге поощряет маленькие, узкофункциональные компоненты и повторное использование кода.

У многих юниксоидов этот подход является второй натурой – маленькие фрагменты кода, решающие конкретные задачи и свободно объединяющиеся. Хитрость, которую мы используем в Python, называется генератором. Это маленький, упрощенный объект, похожий на подпрограмму; его можно также рассматривать как возобновляемую функцию. Лучше пояснить на примере:

```
>>> def fib():
...         a,b = 1,1
...         while 1:
...                 yield a
...                 a,b = b, a+b
...
>>> G = fib()
>>> G
<generator object at 0xb7b59bec>
>>> G.next(), G.next(), G.next(), G.next(), G.next()
(1, 1, 2, 3, 5)
```

Как видите, эта функция при вызове возвращает объект-генератор. Python делает это, поскольку в теле функции есть ключевое слово **yield**, позволяющее неоднократно вызывать метод next генератора – при этом в промежутках между вызовами система вольна использовать процессорное время по своему усмотрению.

Затем поместим его внутрь класса с именем main. Это упрощает взаимодействие с функцией, добавление некоторых метаданных компонента, а в итоге – поддержку таких вещей, как визуальное составление систем с использованием графического компоновщика *Compose*. Простой компонент для вывода содержимого на экран может, например, выглядеть так:

```
from Axon.Component import component
class ConsoleEchoer(component):
        def main(self):
                while 1:
                        while self.dataReady('inbox'):
                                data = self.recv('inbox')
                                print data
                        yield 1
```

Этот код берет данные из inbox (похоже на чтение из **stdin**) и распечатывает их. С другой стороны, я упомянул, что у нас есть и многопоточные компоненты. Предположим, мы хотим написать компонент для чтения с консоли. Он может выглядеть примерно так:

```
from Axon.ThreadedComponent import threadedcomponent
class ConsoleReader(threadedcomponent):
        def main(self):
                while 1:
                        data = raw_input('>>>')
                        self.send(data, 'outbox')
```

По сути, так работает программа чтения мини-консоли нашего «блокнота»: это маленький, специализированный компонент для получения пользовательского ввода. Чтобы соединить эти два элемента в работающую систему, вы создаете канал:

```
from Kamaelia.Chassis.Pipeline import Pipeline
Pipeline(
        ConsoleReader(),
        ConsoleEchoer(),
).run()
```

Функция button в Pygame работает примерно таким же целевым образом – она отрисовывает кнопку, и когда ее нажимают, отсылает сообщение. В качестве последнего примера мы создадим простой инструмент презентации, используя «граф-канал»:

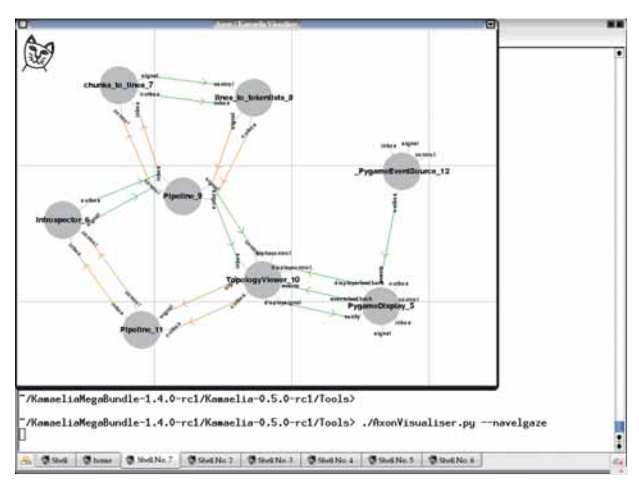
```
Graphline(
        CHOOSER = Chooser(items = files),
        IMAGE = Image(size=(800,600), position=(8,48)),
        NEXT = Button(caption='Next', msg='NEXT', position=(72,8)),
        PREVIOUS = Button(caption='Previous', msg='PREV',
position=(8,8)),
        FIRST = Button(caption='First', msg='FIRST',
position=(256,8)),
        LAST = Button(caption='Last', msg='LAST', position=(320,8)),
        linkages = {
                ('NEXT', 'outbox'):('CHOOSER', 'inbox'),
                ('PREVIOUS', 'outbox'):('CHOOSER', 'inbox'),
                ('FIRST', 'outbox'):('CHOOSER', 'inbox'),
                ('LAST', 'outbox'):('CHOOSER', 'inbox'),
                ('CHOOSER', 'outbox'):('IMAGE', 'inbox'),
        }
).run()
```

Этот код создает четыре кнопки: NEXT, PREVIOUS, FIRST и LAST, которые отсылают сообщение компоненту выбора (CHOOSER). Тот выбирает, какое имя файла отправить компоненту изображения (IMAGE). Затем компонент изображения загружает и отображает картинку.

И более того...

Написать «блокнот» с помощью Kamaelia оказалось намного быстрее, чем на C или C# –наша первая работающая версия была готова за пару дней. Каркас Kamaelia также позволяет сделать приложение намного более гибким, и поскольку оно строится поверх существующих сетевых инструментов, мы получили возможность сосредоточиться исключительно на самом приложении. Конечно, проблемы были, но они возникли бы в любом случае. Этот подход сделал их более явными и более простыми в устранении.

Мы используем Kamaelia для исследования лучших способов доставлять контент BBC, и позволяем BBC работать умнее. Где еще вы могли бы создать PVR с OpenGL и подключить его к IRC-серверу?



› **Визуализатор: средство просмотра работающих систем изнутри.**



Скорая помощь

На сайте Kamaelia в разделе Cookbook можно найти обширную библиотеку примеров. После завершения нашего урока, это лучший способ перейти на следующий этап освоения Kamaelia. http://kamaelia.sourceforge.net/Cookbook.html.

» **Через месяц:** Как сделать документацию XML-способом с DocBook.



Вдохновение по Спарксу

Будучи крупнейшей вещательной компанией в мире, BBC нуждается в серьезных технологиях для сетевого вещания, медиа-кодеках и многом другом. Open source – привлекательное решение, поясняет **Майкл Спаркс**.

Linux Format: Что привело вас в BBC?

Майкл Спаркс: Я закончил Манчестерский университет и устроился работать в Janet Web Cache Service. Мы тогда все расширяли, и делали это хорошо, и это было довольно весело. Использовали Linux, *Squid* и FreeBSD. Была пара машин с Solaris, и пара Irix-машин, но в основном Linux и FreeBSD.

LXF: Это была ваша первая встреча с Linux?

MC: Нет, она произошла в университете. Я изучал информатику, и посреди второго курса решил разобраться, как работают компиляторы, и мне нужен был компилятор для домашнего пользования. Вот я и скачал свободное ПО. Моим первым дистрибутивом был мини-Linux на четырех дисках.

LXF: На дискете?

MC: На четырех! Его нужно было установить на жесткий диск, но он запускался на 386 с 4 МБ памяти, а у меня была как раз такая машина. Оттуда все и пошло. Я и понравился Janet Web Cache Service потому, что имел опыт работы в Linux. Потом я стал подумывать о переезде в Лондон, но пока жил в Манчестере. Компания Inktomi тогда набирала сотрудников, и вербовщик нашел мое имя в списке рассылки *Squid*, так что благодаря *Squid* я попал в Inktomi. Они занимались кэшированием крупных сетей, сетевыми системами и сетями доставки мультимедиа.

LXF: А это, в свою очередь, было нужно BBC?

MC: BBC в то время нанимала инженеров-исследователей. И так совпало – они об этом не говорили, пока меня не наняли – они создавали новую исследовательскую группу. До этого исследовательский отдел состоял из четырех основных групп: группа Studio занималась производством; Transmission – вещанием; одна группа отвечала за размещение передатчиков в Великобритании для получения лучшего охвата; и еще одна для систем внутри BBC. Но для сетевых исследований не было ничего; они собрались основать новую группу, и мне посчастливилось присоединиться к ней вовремя. Вот почему это область моих исследований.

LXF: Вы управляли этим процессом, или что-то советовали?

MC: Нет, не советовал; все произошло случайно. BBC была в процессе формирования этого маленького сетевого подразделения, человека на четыре-пять.

LXF: Его задачей было распределение контента в сети?

MC: Да, и они понимали, что нужны некоторые исследования в этой области, потому что все это [раньше] было спонтанно. Многое на сайте BBC было в стиле: «Стянем одно отсюда, другое – оттуда, заставим заработать, а как разослать, неважно». А я как раз в этом разбирался, потому что занимался этим несколько предыдущих лет. Они спросили: «Как нам это рассылать?», ну, я и говорю: «Что ж, по-моему, вопрос интересный, я не ожидал, что буду делать это».

Довольно быстро выяснилось, что потоковое вещание само по себе жутко дорогое. Сервер Real Networks тогда еще не был открыт. Они перешли на по-процессорную модель, но процессор так и так может обработать лишь определенное число потоков, и это был просто другой способ описать ту же проблему. В наши дни выбор богаче, а тогда было примерно так: «Нужно разработать поточный сервер, потому что нам надо обрабатывать 20 миллионов параллельных потоков, и это будет дешевле, чем оплачивать серверную нагрузку».

LXF: Насколько это было похоже на вашу прежнюю работу?

MC: Что интересно, в Inktomi продавались системы web-кэширования. Эти решения поддерживали также потоковое кэширование и включали сети распространения мультимедиа. Таким образом, штуки вроде сетей распространения использовали групповое вещание на уровне приложений, то есть отдельных аудио- и видеопакетов, а не IP-пакетов, как это обычно бывает. Источники и отправители также основывались на контенте, а не на IP, что было шагом вперед и значительно упрощало жизнь.

Так что у меня был большой опыт в этой области, а поскольку мы продавали эти вещи, я должен был довольно подробно знать их внутренности и как все структурировано. Кроме того, я разбирался в *Squid*, то есть понимал, как двигаться вперед.

О KAMAELIA

«Фактически я решаю задачу, как сделать параллельные системы удобными для обычных людей.»

LXF: Вы не поясните, как Kamaelia работает в техническом плане?

MC: Для моделирования параллелизма мы используем генераторы Python. Потому что, на самом деле, это дает нам возобновляемые функции. Так что вы можете написать нечто, на вид однопоточное, и запускать этот поток снова и снова.

LXF: А они распараллеливаются?

MC: Вы можете распараллелить их. Фактически, все они последовательны, потому что это куски кода, которые работают, пока не наткнутся на точку результата [yield]. Генератор Python – это небольшая функция со словом 'yield' в теле. Если в ней есть yield, то при ее вызове возвращается объект, и вы можете вызвать его метод next, и он будет выполняться до тех пор, пока не наткнется на следующий yield, а тот – на следующий, и так все время, пока вы запускаете next. Так что next проходит до следующего yield, потом следующего, и т.д. Очевидно, вы можете создать их множество, вот и выходит параллелизм. Можно считать это альтернативой конечным автоматам, но замечательно, что это не шиворот-навыворот, это правильный способ.

Например, если вам нужно много интересных обработок ошибок, обработка ошибок все еще хороша и понятна – каждый знает, что нечто однопоточное написать легко, здесь нет серьезных проблем. Вам не надо думать: «Ладно, я остановлю это, и что я должен сделать, как сохранить это состояние, и как вернуться в эту позицию?». Не надо предусматривать откаты и тому подобное.

LXF: То есть Python для Kamaelia – единственный выбор, или можно использовать что-то еще?

MC: С выбором Python получилось забавно. Первоначально предполагалась команда из трех разработчиков: Тима Борера [Tim Borer], которого вы, наверное, знаете по проекту Dirac, Джозефа Лорда [Joseph Lord] и меня. А кончилось тем, что на следующую пару лет остался, по сути, один я, поскольку Джозефа утащили на какую-то другую работу, а Тим занялся проектом Dirac. Сейчас со мной еще работает Мэтт Хэммонд [Matt Hammond].

Но главное, мы хотели выбрать язык, на котором никто из нас не писал. В исследовательской организации хочется что-то изучать. А как вы учитесь? Вы беретесь за то, чего раньше не делали. И я подумал: «Ну, раз он поддерживает идею возобновляемых функций, берем, потому что в сущности моя задача – сделать программы доступными для сопровождения. Неважно, трудно ли на нем писать». Kamaelia упрощает разработку программ – в первую очередь потому, что все выглядит однопоточным, но фактически я хочу сделать эти параллельные системы обслуживаемыми обычными людьми.

Обычно параллельные системы пишутся [людьми, говорящими]: «О, мы собираемся посадить на эту работу наших лучших программистов», и так происходит везде. Выпустим продукт как можно быстрее – посадим на него лучших программистов, и они его сделают, а затем передадим эксплуатационникам, которые по определению менее квалифицированны: будь это не так, они, вероятно, работали бы над следующим проектом. Примерно так работают многие организации.

Таким образом, меня действительно беспокоило, как заставить работать параллельные системы для основной массы пользователей, а не для самых квалифицированных. Вот почему я хотел, чтобы все выглядело однопоточным, потому что, прежде всего, вы можете написать свой кусок однопоточным, убедиться, что он работает, нашлепать несколько директив yield в произвольных местах, и это станет компонентом. И затем вы сможете повторно использовать его.

LXF: Это действительно хороший аргумент. Так вы не знали, что генераторы предоставляют такую функциональность?

MC: Я такую функциональность искал, обычно это называется «cur-routines». Ну, а в Python – генераторами. Они не столь общие, как стандартные сервисные подпрограммы, и это их большое преимущество, поскольку оно гарантирует, что все будет намного компактнее. А чем они компактнее, тем более обстоятельные, а еще – их легче использовать повторно.

LXF: BBC понимала необходимость открыть исходный код Kamaelia и создать вокруг своего рода сообщество?

MC: Некоторые понимали. BBC не однородна, это стоит помнить. BBC известна своей приверженностью открытым стандартам. Например, хорошо известно, что если у нас есть открытый стандарт, мы найдем группу людей, которые придут и его поддержат, и что-нибудь для него сделают.

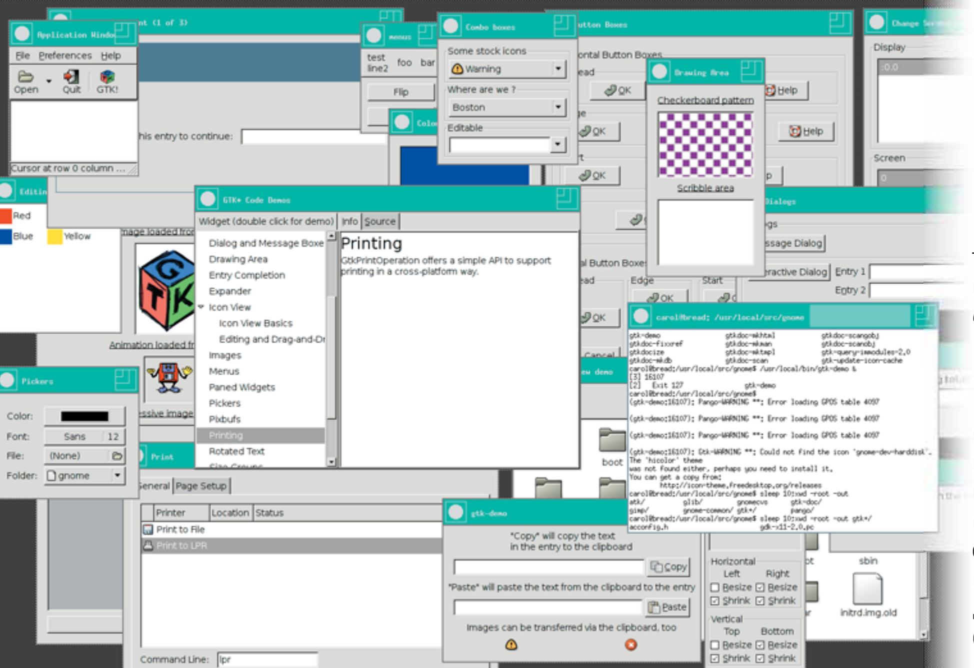
Естественно, BBC надеется на изготовителей оборудования, которые придут и построят что-то вокруг открытого стандарта на оборудование – чтобы люди о нем даже не задумывались. Но идея, что открытие ПО дает, потенциально, наличие сообщества, на некоторых уровнях им плохо знакома.

BBC очень изменилась. В смысле, легко сказать «BBC», а BBC-то, на самом деле, это около 27 тысяч человек. А еще есть люди, которые живут и дышат Open Source. LXF





ЧАСТЬ 1: Прочитав нашу серию статей о Qt/KDE, вы укрепились во мнении, что о лучшем и мечтать нельзя, и уже готовы удивить мир своей разработкой? Выслушайте сначала контр-аргументы – **Андрей Боровский** предлагает вашему вниманию новую серию учебников о GTK+!



Вопреки распространенному убеждению, эти библиотеки можно использовать в коммерческих проектах. Даже Билл Гейтс смог бы воспользоваться ими!

Руководство программиста FLTK 1.1.7

Линус любит KDE. Он сам неоднократно говорил об этом, не стесняясь в выражениях по адресу GNOME. Я тоже не большой поклонник GNOME, а вот инструментарий *GTK+* мне очень нравится. Думаю, в таком сочетании предпочтений нет ничего особенного, ведь *GTK+* – это далеко не только GNOME. Даже если вы – фанатичный пользователь KDE, вы наверняка время от времени работаете в редакторе *Gimp*, основанном на библиотеках *GTK+*. Собственно, аббревиатура *GTK+* и расшифровывается как *Gimp ToolKit* – «набор инструментов для *Gimp*).

Сегодня мы начинаем серию статей, посвященных *GTK+*. Большая часть приложений, которые мы напишем, будет работать независимо от среды GNOME, но мы не оставим без внимания и взаимодействие с этой популярной оболочкой. Скорее всего, вы, уважаемый читатель Linux Format, уже представляете себе, что такое *GTK+*. Тем не менее, следует все-таки напомнить, что *GTK+* представляет собой открытый набор графических компонентов, предназначенных для создания приложений на платформах Linux/Unix, Win32 и MacOS. *GTK+* включает в себя большое количество визуальных элементов, используемых при создании графического интерфейса приложений, а также некоторые вспомогательные невизуальные элементы. На *GTK+* основаны такие приложения, как графическая среда GNOME, редактор растровой графики *Gimp*, текстовый редактор *AbiWord*, табличный процессор *Gnumeric* и многие-многие другие. Помимо библиотек, реализующих различные элементы графического интерфейса приложений, *GTK+* снабжен вспомогательными утилитами. *Glade*, например, позволяет проектировать интерфейсы *GTK*-приложений в режиме визуального редактирования. Если вы установили пакеты разработки *GTK+*/GNOME, то в вашей системе наверняка также установлена программа *Devhelp*, которая представляет собой браузер документации программиста по *GTK+*, GNOME, *Gimp* и *Evolution*.

Можно ли назвать *GTK+* предпочтительным инструментарием разработчика графических приложений? Вообще говоря, если вы выбираете набор инструментов для создания графического интерфейса нового Linux-приложения, вы не должны особенно беспокоиться о том, какую оболочку предпочитают ваши пользователи. В соответствии с идеологией свободного выбора, которая пронизывает Linux, различные платформы и компоненты системы очень хорошо уживаются между собой. Средства взаимодействия между приложениями, использующими разные графические компоненты, тоже быстро совершенствуются. Для того чтобы сделать выбор между *GTK+* и *Qt*/KDE, следует хотя бы бегло ознакомиться с возможностями и особенностями каждого инструментария.

Как это нередко бывает в мире открытого ПО, в настоящее время активно используются сразу две ветки *GTK+*. Многие разработчики приложений, воспользовавшиеся в свое время *GTK+* 1.2, не видят необходимости переходить на новые версии пакета, поэтому *GTK+* 1.2 все еще можно встретить во многих дистрибутивах Linux. Новые (на момент написания этой статьи) приложения используют *GTK+* версий 2.x. Примеры из нашей серии статей должны работать со всеми версиями *GTK+*, начиная с 2.0, если иное не указано явно.

Наши инструменты

Приложения *GTK+* под Linux – это, прежде всего, приложения Linux. Для создания приложений *GTK+* вам понадобятся стандартные инструменты разработчика – *GCC* и *automake* со товарищи. Помимо этого, в вашей системе должны быть установлены пакеты *GTK+-devel**, *atk-devel**, *pango-devel** со всеми зависимостями. Кроме того, рекомендуем установить пакеты *libgnome-devel* и *glade*. После установки пакетов разработчика в нашем распоряжении окажутся утилиты командной строки *glib-config* и *pkg-config* (в некоторых системах также может быть установлена утилита *GTK-config*). Эти утилиты выводят информацию о расположении базовых библиотек *GTK+* в вашей системе. При этом утилита *glib-config* предназначена исключительно для вывода информации о библиотеке *glib*, а *pkg-config* выводит информацию о самых разных библиотеках. Если, например, скомандовать в окне консоли

```
pkg-config --libs GTK+-2.0
```

вы увидите нечто вроде

```
-L/usr/X11R6/lib -L/opt/gnome/lib -lGTK-x11-2.0 -lgdk-x11-2.0 -latk-1.0 -lgdk_pixbuf-2.0
```



знакомство

```
-lpangocairo-1.0 -lpango-1.0 -lcairo -lgobject-2.0 -lgmodule-2.0 -ldl -lglib-2.0
-lfreetype -lfontconfig -lXrender -lX11 -lXext -lpng12 -lz -lglitz -lm
```

Вывод команды представляет собой список ключей компоновщика GCC, которые необходимо указать для подключения к основанному на *GTK+ 2.x* приложению всех необходимых ему библиотек. Как вы, конечно, догадались, этот список сгенерирован не столько для того, чтобы удовлетворить наше любопытство, сколько для автоматизации работы компоновщика в процессе сборки приложений *GTK+*, что мы увидим ниже. Команда

```
pkg-config --cflags GTK+-2.0
```

выдаст все ключи компилятора, необходимые для компиляции приложения *GTK+ 2.x*.

Hello GTK+ World!

Теперь, когда мы знаем, что нам нужно для того, чтобы скомпилировать приложение *GTK+*, мы готовы написать простейшую программу. Наша первая программа (назовем ее *helloworld*) создает простое окно с кнопкой. Щелчок по кнопке приводит к закрытию окна и завершению работы программы. Рассмотрим исходный текст программы *helloworld* (этот исходный текст вы найдете на диске в файле **helloworld.c**).

```
#include <GTK/GTK.h>
static void button_clicked(GTKWidget * widget, gpointer data)
{
  g_print("Button was clicked!\n");
}
static gboolean delete_event(GTKWidget * widget, GdkEvent * event,
```

»

GTK+ или Qt/KDE?

Сделать выбор между двумя популярными графическими инструментариями, которые долгое время конкурировали друг с другом и потому многое друг у друга переняли, непросто. Критерием истины здесь может быть только практика разработки вашего проекта, однако, мы попробуем провести некоторый формальный анализ. Сводка важнейших параметров *GTK+* и *Qt* приводится в следующей таблице.

Функция	GTK+	Qt
Базовый интерфейс	C	C++
Лицензия	LGPL	Двойная
Порт для Win32 и MacOS	+	+
Возможность прямых вызовов из C	+	-
Интерфейсы Java, Perl, Python	+	+
Порт для .NET	+ (GTK#)	+ (Qt#)
Бесплатно для коммерческого использования	+	-

То, что по многим параметрам и *GTK+*, и *Qt* выставлены плюсы, не означает, что соответствующие возможности этих платформ равноценны. Например, порты *GTK#* и *Qt#* на данный момент отличаются довольно сильно: первый проект поддерживается командой разработчиков Mono и, ввиду незавершенности Mono-реализации Windows.Forms, является в нем де-факто стандартом для создания GUI; второй является любительским и не видел обновлений аж с 2002 года! С Java ситуация в какой-то мере обратная: привязки *Qt Jambi* разрабатываются непосредственно Trolltech, тогда как интеграция *GTK+* и Java выполняется в рамках стороннего проекта Java-GNOME, который хоть и включает в себя интерфейсы Java для всех базовых компонентов *GTK+*, но ориентирован прежде всего на GNOME. Python одинаково хорошо работает и с *Qt*, и *GTK+*, а PerlQt не обновлялся с 2003 года и не поддерживает *Qt4*. В целом, можно сказать, что *GTK+* выигрывает в поддержке «неродных» и скриптовых языков – по крайней мере, «по очкам».

Остановимся на том, что важно для любого программиста – на лицензировании. *GTK+* распространяется на условиях LGPL. Иначе говоря, его можно использовать совершенно бесплатно и для создания открытых приложений, и для коммерческих разработок (код приложения может быть закрытым, если он не является расширением самого набора *GTK+*). С *Qt* ситуация иная. Вы можете пользоваться *Qt* бесплатно для создания открытых программ, но за лицензию, позволяющую разрабатывать коммерческие продукты, придется платить, причем немало.

С кросс-платформенностью тоже не все так просто, как может показаться на первый взгляд. Большинство *Qt*-ориентированных Linux-проектов использует не только *Qt*, но и дополнительную функциональность, которую предоставляют библиотеки KDE. В то же время в тандеме *Qt*/KDE по-настоящему кросс-платформенной является только *Qt* (ситуация изменится после выхода KDE4, но последняя не доросла еще даже до бета-версии). Из этого следует, что если вы намереваетесь создавать кросс-платформенный продукт и делаете свой выбор в пользу Qt/KDE, вам придется ограничиться возможностями *Qt*. В пользу «сладкой парочки» Qt/KDE следует сказать, однако, что, по мнению многих разработчиков, основанный на C++ интерфейс *Qt*/KDE проще и компактнее, чем ориентированный на C интерфейс *GTK+*. Кроме того, *Qt* гораздо аккуратнее интегрирует ваше приложение с окружающей средой: *Skype* в Windows выглядит и ведет себя в точности как приложение Windows, и немногие догадывались, что *Google Earth* использует *Qt* (а не, скажем, MFC) до выхода Linux-версии. Приложения *GTK+* тоже можно собрать и запустить в Windows и Mac OS – но не ждите особой эстетики и безупречного поведения.

Важно также понимать, что как *Qt*/KDE, так и *GTK+* используют при построении интерфейса принципы объектно-ориентированного программирования. Разница между двумя наборами инструментов заключается в том, что объектно-ориентированная модель *GTK+* реализована «добровольно» в рамках интерфейса C, тогда как интерфейс программирования *Qt*/KDE закован в объектно-ориентированные конструкции C++. Функции интерфейса *GTK+* могут быть вызваны из программ, написанных на языках, которые не поддерживают импорт классов C++ в формате GCC. Для того, чтобы не объектно-ориентированные языки могли импортировать интерфейс *Qt*/KDE, приходится создавать громоздкие комплексы функций-оберток, «переводящих» вызовы методов классов C++ в формат C.

› Угловатые, «неродные» виджеты не дают *Inkscape* почувствовать себя в Windows, как дома.

»



```
  gpointer data)
{
  g_print("Delete event occurred\n");
  return FALSE;
}
static void destroy(GTKWidget * widget, gpointer data)
{
  g_print("Destroy signal was sent\n");
  GTK_main_quit();
}
int main(int argc, char ** argv)
{
  GTKWidget * window;
  GTKWidget * button;
  const gchar * title = "Hello World!";
  GTK_init(&argc, &argv);
  window = GTK_window_new(GTK_WINDOW_TOPLEVEL);
  GTK_window_set_title(GTK_WINDOW(window), title);
  GTK_container_set_border_width(GTK_CONTAINER(window), 10);
  g_signal_connect(G_OBJECT(window), "delete_event",
    G_CALLBACK(delete_event), NULL);
  g_signal_connect(G_OBJECT(window), "destroy",
    G_CALLBACK(destroy), NULL);
  button = GTK_button_new_with_label("Quit");
  g_signal_connect(G_OBJECT(button), "clicked",
    G_CALLBACK(button_clicked), NULL);
  g_signal_connect_swapped(G_OBJECT(button), "clicked",
    G_CALLBACK(GTK_widget_destroy), G_OBJECT(window));
  GTK_container_add(GTK_CONTAINER(window), button);
  GTK_widget_show(button);
  GTK_widget_show(window);
  GTK_main();
  return 0;
}
```

Прежде чем мы пройдем нашу программу шаг за шагом, необходимо дать некоторые пояснения общего характера. В графических интерфейсах, реализованных на объектно-ориентированных языках, все визуальные элементы представляются классами. В интерфейсе *GTK+*, реализованном на языке C, визуальным элементам соответствуют структуры данных. Эти структуры (мы будем называть их объектами) группируются в иерархии, которые соответствуют отношениям объектов интерфейса. Следует отметить, что понятие «иерархии» здесь достаточно условное, отношения объектов *GTK+* не следует путать с иерархическими отношениями классов в объектно-ориентированных языках.

Корнем иерархии объектов *GTK+* является абстрактный объект GObject. Ниже в иерархической лестнице расположен объект GTKObject, потомком которого является объект GtkWidget, который, в свою очередь, служит корнем иерархии всех визуальных элементов (виджетов). Здесь уместно сказать несколько слов и о формировании идентификаторов в *GTK+*. Как и во многих интерфейсах Unix, имена функций, типов данных, констант и макросов в *GTK+* начинаются с префикса, указывающего на имя библиотеки, которая экспортирует данный идентификатор. Имена функций, экспортируемых библиотекой *GTK* (*libGTK*), начинаются с префикса gtk_, имена типов данных из этой библиотеки предваряются префиксом GTK, а имена констант и макросов имеют префикс GTK_. Имена функций, типов и констант с макросами, экспортируемых библиотекой *GLib* (*libglib*), начинаются с префиксов g_, g и G_, соответственно.

Текст нашей программы *helloworld* начинается с определения трех статических функций. Эти функции представляют собой обработчики сигналов *GTK+*. Как и все современные многооконные графические системы, *GTK+* базируется на событийно-управляемой архитектуре. Когда в графической системе происходит нечто, связанное с одним из окон приложения (щелчок мышью, нажатие на клавишу, сокрытие окном другого приложения – то есть возникает необходимость перерисовки), данному окну посылается сообщение. *GTK+* преобразует сообщение оконной системы в сигнал и вызывает функцию-обработчик этого сигнала. В качестве аргументов функции-обработчику передаются данные об объекте-источнике и параметрах события. Механизм сигналов абстрагирован от механизма сообщений низкоуровневой графической подсистемы и отражает скорее структуру *GTK+*, нежели структуру системы низкого уровня. Источником сигналов, связанных с визуальным элементом управления, в *GTK+* считается сам визуальный элемент. Для обработки сигналов *GTK+* использует функции обратного вызова (callback). *Qt*, кстати, тоже использует функции обратного вызова, скрытые под надстройкой сигналов и слотов. Похоже, что прогрессивное человечество, по крайней мере, та его часть, которая пишет на C и C++, ничего лучшего пока не придумало. Обработчики сигналов представляют собой обычные функции C. Например, если мы создадим функцию-обработчик button_clicked и свяжем ее с сигналом clicked визуального элемента-кнопки, обработчик button_clicked будет вызываться в ответ на щелчок мышью по кнопке. Мы можем связать один обработчик с несколькими сигналами и назначить одному сигналу несколько обработчиков. Первым параметром функции обработчика должен быть указатель на объект-источник сигнала, вторым параметром – указатель на произвольную структуру данных, которую программист может связать с данным сигналом. Помимо сигналов и *GTK+* определены события, которые соответствуют событиям низкоуровневой системы X Window. По сути, события – это те же сигналы, но функции-обработчики событий отличаются от обработчиков обычных сигналов списком параметров. Имена событий имеют окончание _event. В нашей программе функция button_clicked() является обработчиком сигнала, а функция delete_event() – обработчиком события. Вы можете видеть, что списки параметров этих функций различаются.

Теперь мы можем более подробно описать каждую функцию обратного вызова, определенную в нашей программе. Функция button_clicked – это обработчик сигнала clicked кнопки приложения. Функция delete_event обрабатывает событие delete_event, а функция destroy представляет собой обработчик сигнала destroy. Событие delete_event генерируется системой X Window в случае, если пользователь пытается закрыть окно приложения. Сигнал destroy всегда посылается приложению *GTK+* во время завершения его работы. Действия, выполняемые функцией main() нашей программы, можно разделить на шесть стадий: создание и настройка главного окна, назначение обработчиков сигналов и событий окна, создание кнопки, назначение обработчиков сигнала clicked кнопки, расположение кнопки в главном окне, отображение окна и кнопки.

Работа программы начинается с вызова функции gtk_init(). Как следует из названия, gtk_init() инициализирует приложение (устанавливает значения параметров *GTK+*) и обрабатывает специальные аргументы командной строки, которые могут быть переданы приложению *GTK+*.

Далее мы создаем главное окно приложения с помощью функции gtk_window_new(). Единственный параметр функции указывает, что мы создаем обычное главное окно. Другой возможный параметр – GTK_WINDOW_POPUP позволяет создать всплывающее окно (popup window). Функция gtk_window_new() возвращает указатель на структуру GTKWidget, соответствующую созданному окну. Этот указатель

мы сохраняем в переменной window. Функция gtk_window_set_title() устанавливает надпись в заголовке окна. Первым аргументом функции должен быть идентификатор окна, вторым аргументом – текст заголовка. Функция gtk_window_set_title() ожидает, что первым аргументом будет значение типа GTKWindow * (указатель на объект-окно). Однако, поскольку переменная window имеет тип указатель на GTKWidget, мы выполняем приведение типа с помощью макроса GTK_WINDOW. Приведение типов в данном случае необязательно, и мы выполняем его только для того, чтобы избавиться от предупреждений, выдаваемых компилятором. Главное окно приложения, как и многие другие объекты *GTK+*, представляет собой контейнер. Объекты-контейнеры могут содержать произвольное количество дочерних визуальных элементов. При этом контейнеры обычно управляют расположением и размером дочерних виджетов, чем очень облегчают жизнь программиста (более подробное знакомство с контейнерами состоится в одной из следующих статей серии). Функция gtk_container_set_border_width() устанавливает ширину границы контейнера. Для главного окна «ширина границы» означает расстояние от края дочернего элемента до края окна.

Функция g_signal_connect() связывает сигнал объекта с обработчиком. Первый параметр функции – объект-источник сигнала. Мы приводим идентификатор объекта к типу GObject с помощью макроса G_OBJECT. Второй параметр функции – строка с именем сигнала. Третий параметр – это указатель на функцию-обработчик (мы приводим его значение к типу GCallback). Последний параметр функции g_signal_connect() представляет собой указатель на произвольную структуру данных, которую мы можем передать обработчику события. Для большинства обработчиков событий мы будем оставлять это значение равным **null**. Мы назначаем обработчики двум другим сигналам окна – delete_event и destroy (для удобства имена функций-обработчиков выбраны аналогичными именам сигнала).

Теперь нам необходимо создать кнопку, для чего мы пользуемся функцией gtk_button_new_with_label(). С помощью этой функции, чье имя говорит само за себя, мы создаем кнопку и указываем надпись на ней. Единственный параметр функции gtk_button_new_with_label() – это текст надписи на кнопке. Возвращаемое функцией значение представляет собой указатель на объект GTKWidget, соответствующий созданной кнопке. Почему функции, создающие окно и кнопку, возвращают указатели на объекты GTKWidget, а не на соответствующие этим визуальным элементам объекты GTKWindow и GTKButton (оба типа определены в библиотеке *GTK*)? Вероятно, это сделано для того, чтобы избавить нас от лишних операций приведения типов. Ведь большая часть функций, работающих с визуальными элементами, принимает в качестве параметра именно указатель на GTKWidget.

Кнопка может быть источником нескольких сигналов, важнейший из которых, сигнал clicked, оповещает приложение о том, что по кнопке щелкнули мышью (или выполнили аналогичное действие с помощью клавиатуры). Мы назначаем сигналу clicked кнопки button два обработчика событий – определенный нами обработчик button_clicked() и функцию gtk_widget_destroy(), которая завершает работу программы, посылая при этом сигнал destroy. Любопытно отметить, что функция gtk_widget_destroy(), вообще говоря, не является обработчиком сигнала. Возможность использовать «простые» функции *GTK+* в качестве обработчиков сигналов представляет собой полезную особенность API *GTK+*, с которой мы еще встретимся не один раз. Для назначения обработчика gtk_widget_destroy() мы используем функцию g_signal_connect_swapped(). Эта функция аналогична функции g_signal_connect(), за исключением того, что при вызове обработчика параметры передаются ему в другом порядке (подробнее об использовании g_signal_connect_swapped() мы поговорим в следующих статьях).

Функция gtk_container_add() добавляет дочерний элемент в контейнер. Первый параметр функции – это указатель на объект-контейнер, вторым параметром должен быть указатель на добавляемый объект. Мы используем функцию gtk_container_add() для того, чтобы разместить кнопку в окне. И кнопка, и окно примут при этом разумные размеры (определяемые длиной надписи на кнопке и шириной границы окна-контейнера). Наконец, мы выполняем два вызова gtk_widget_show(), делающих визуальные элементы интерфейса (кнопку и окно) видимыми. Последняя функция, которую мы вызываем явным образом – gtk_main(), она запускает цикл обработки сообщений. Теперь наша программа сможет реагировать на сигналы, посылаемые визуальными элементами управления.

Нам осталось рассмотреть обработчики сигналов. Обработчик button_clicked() распечатывает в окне терминала сообщение о том, что кнопка была нажата. Для этого используется функция g_print(), которая работает аналогично функции printf(). Сигнал-событие delete_event посылается программе, как уже говорилось, при попытке закрыть окно приложения. Обработчик этого события может отменить завершение работы приложения (для этого, как ни странно, он должен вернуть значение **true**, а не **false**). Сигнал destroy сообщает приложению, что его работа будет завершена и не допускает отмены завершения. Если вы закрываете окно приложения, щелкая по кнопке в заголовке окна, приложение сначала получит сигнал delete_event, а затем, если обработчик этого сигнала вернет значение FALSE, сигнал destroy. Функция gtk_widget_destroy() (связанная в нашем примере с кнопкой button) посылает только сигнал destroy, но не delete_event. Хотя получение сигнала destroy в нашей программе свидетельствует об уничтожении главного визуального элемента (и его дочерних элементов), само по себе это уничтожение не приводит к выходу из цикла обработки сообщений. Выход из цикла нужно выполнить явным образом с помощью функции gtk_main_quit().

Для того, чтобы скомпилировать наш пример, мы воспользуемся уже известной нам утилитой *pkg-config*. Команда компиляции может выглядеть так:

```
gcc -Wall helloworld.c -o helloworld `pkg-config --cflags GTK+-2.0`
`pkg-config --libs GTK+-2.0`
```

Мы вставляем в командную строку ключи, которые выдает нам утилита *pkg-config*. Теперь мы можем, наконец, полюбоваться творением наших рук (**рис. 1**).

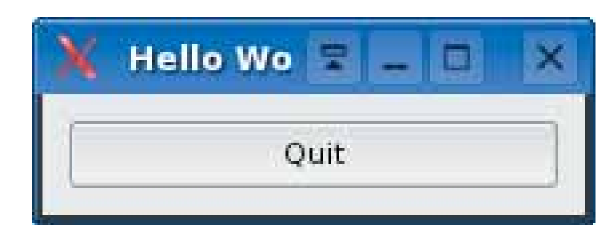


Рисунок 1. Наше первое приложение GTK+.

Приступая к изучению нового интерфейса, очень приятно быстро получить первую работающую программу. Еще приятнее понимать, как это программа работает. Углубленному изучению механизма сигналов и взаимодействия визуальных компонентов *GTK+* будет посвящена следующая статья. LXF

Через месяц Мы займемся углубленным изучением механизма сигналов GTK+.

Потоки: создание

ЧАСТЬ 7: Сегодня **Андрей Боровский** намерен рассказать о жизненном цикле потоков – от создания до принудительного завершения.

“Processes are like human beings... A small difference is that sex is not really common among processes as each process has just one parent.”

Understanding the Linux Kernel, 3rd Edition
By Daniel P. Bovet, Marco Cesati



Многопоточность является естественным продолжением многозадачности, точно так же как виртуальные машины, позволяющие запускать несколько ОС на одном компьютере, представляют собой логическое развитие концепции разделения ресурсов. В рамках неформального, но простого, определения, поток – это последовательность выполнения машинных инструкций. В многопоточном приложении одновременно работает несколько потоков. Некоторые авторы избегают термина «поток» и используют вместо него термин «нить» (от англ. *«thread»*), вероятно для того, чтобы потоки программ не путались с потоками ввода-вывода. Для обозначения последовательного выполнения цепочки инструкций мне лично больше нравится термин «поток», которым я и буду пользоваться. Надеюсь, читатели Linux Format не запутаются в контекстах и, встретив слово поток, всегда поймут, идет ли речь о потоках программы, потоках ввода вывода, или о бурных паводковых потоках.

Прежде чем приступать к программированию потоков, следует ответить на вопрос: а нужны ли они вам. Мы уже знаем, насколько хорошо развиты в Linux средства межпроцессного взаимодействия. С помощью управления процессами в Linux можно решить многие задачи, которые в других ОС решаются только с помощью потоков. Потоки часто становятся источниками программных ошибок особого рода. Эти ошибки возникают при использовании потоками разделяемых ресурсов системы (например, общего адресного пространства) и являются частным случаем более широкого класса ошибок – ошибок синхронизации. Если задача разделена между независимыми процессами, то доступом к их общим ресурсам управляет операционная система, и вероятность ошибок из-за конфликтов доступа снижается. Впрочем, разделение задачи между несколькими независимыми процессами само по себе не защитит вас от других разновидностей ошибок синхронизации. В пользу потоков можно указать то, что накладные расходы на создание нового потока в многопоточном приложении обычно ниже, чем накладные расходы на создание нового самостоятельного процесса. Уровень контроля над потоками в многопоточном приложении выше, чем уровень контроля приложения над дочерними процессами. Кроме того, многопоточные программы не склонны оставлять за собой вереницы зомби или «осиротевших» независимых процессов.

Первая подсистема потоков в Linux появилась около 1996 года и называлась без лишних затей – LinuxThreads. Рудимент этой подсистемы, который вы найдете в любой современной системе Linux, – файл **/usr/include/pthread.h**, указывает год выпуска – 1996 и имя разработчика – Ксавье Лерой (Xavier Leroy). Библиотека LinuxThreads была попыткой организовать поддержку потоков в Linux в то время, когда ядро системы еще не предоставляло никаких специальных механизмов для работы с ними. Позднее разработку потоков для Linux вели сразу две конкурирующие группы – NGPT и NPTL. В 2002 году группа NGPT фактически присоединилась к NPTL, и теперь реализация потоков NPTL является стандартом Linux. Подсистема потоков Linux стремится соответствовать требованиям стандартов POSIX, так что новые многопоточные приложения Linux должны без проблем компилироваться на других POSIX-совместимых системах.

Потоки и процессы

Тем, кто впервые познакомился с концепцией потоков, изучая программирование для Windows, модель потоков Linux покажется непривычной. В среде Microsoft Windows процесс – это контейнер для потоков (именно этими словами о процессах говорит Джеффри Рихтер в своей

Месяц назад Мы изучали процессы и потоки в современных ОС.



и уничтожение

классической книге «Программирование приложений для Microsoft Windows»). Процесс-контейнер содержит как минимум один поток. Если потоков в процессе несколько, приложение (процесс) становится многопоточным. В мире Linux все выглядит иначе. В Linux каждый поток является процессом, и для того, чтобы создать новый поток, нужно создать новый процесс. В чем же, в таком случае, заключается преимущество многопоточности Linux перед многопроцессностью? В многопоточных приложениях Linux для создания дополнительных потоков используются процессы особого типа. Эти процессы представляют собой обычные дочерние процессы главного процесса, но они разделяют с главным процессом адресное пространство, файловые дескрипторы и обработчики сигналов. Для обозначения процессов этого типа, применяется специальный термин – легкие процессы (*lightweight processes*). Прилагательное «легкий» в названии процессов-потоков вполне оправдано. Поскольку этим процессам не нужно создавать собственную копию адресного пространства (и других ресурсов) своего процесса-родителя, создание нового легкого процесса требует значительно меньших затрат, чем создание полновесного дочернего процесса. Поскольку потоки Linux на самом деле представляют собой процессы, в мире Linux нельзя говорить, что один процесс содержит несколько потоков. Если вы скажете это, в вас тут же заподозрят вражеского лазутчика!

Интересно рассмотреть механизм, с помощью которого Linux решает проблему идентификаторов процессов-потоков. В Linux у каждого процесса есть идентификатор. Есть он, естественно, и у процессов-потоков. С другой стороны, спецификация POSIX 1003.1c требует, чтобы все потоки многопоточного приложения имели один идентификатор. Вызвано это требование тем, что для многих функций системы многопоточное приложение должно представляться как один процесс с одним идентификатором. Проблема единого идентификатора решается в Linux весьма элегантно. Процессы многопоточного приложения группируются в группы потоков (*thread groups*). Группе присваивается идентификатор, соответствующий идентификатору первого процесса многопоточного приложения. Именно этот идентификатор группы потоков используется при «общении» с многопоточным приложением. Функция getpid(2) возвращает значение идентификатора группы потока, независимо от того, из какого потока она вызвана. Функции kill(), waitpid() и им подобные по умолчанию также используют идентификаторы групп потоков, а не отдельных процессов. Вам вряд ли понадобится узнавать собственный идентификатор процесса-потока, но если вы захотите это сделать, вам придется воспользоваться довольно экзотичной конструкцией. Получить идентификатор потока (thread ID) можно с помощью функции gettid(2), однако саму функцию нужно еще определить с помощью макроса _syscall [это само по себе уже является явным намеком на то, что вам не следует усердствовать с ее использованием, – прим. ред.]. Работа с функцией gettid() выглядит примерно так:

```
#include <sys/types.h>
#include <linux/unistd.h>
...
_syscall0(pid_t,gettid)
...
pid_t my_tid;
my_tid = gettid();
```

Более подробную информацию вы можете получить на страницах man, посвященных gettid() и _syscall.

Потоки создаются функцией pthread_create(3), определенной в заголовочном файле **pthread.h**. Первый параметр этой функции представляет собой указатель на переменную типа pthread_t, которая служит идентификатором создаваемого потока. Второй параметр, указатель на переменную типа pthread_attr_t, используется для передачи атрибутов потока. Третьим параметром функции pthread_create() должен быть адрес функции потока. Эта функция играет для потока ту же роль, что функция main() для главной программы. Четвертый параметр функции pthread_create() имеет тип void *. Этот параметр может использоваться для передачи функции потока произвольного аргумента. Вскоре после вызова pthread_create() функция потока будет запущена на выполнение параллельно с другими потоками программы. Таким образом, собственно, и создается новый поток. Я говорю, что новый поток запускается «вскоре» после вызова pthread_create() потому, что перед тем как запустить новую функцию потока, нужно выполнить некоторые подготовительные действия, а поток-родитель, между тем, продолжает выполняться. Непонимание этого факта может привести вас к ошибкам, которые будет трудно обнаружить. Если в ходе создания потока возникла ошибка, функция pthread_create() возвращает ненулевое значение, соответствующее номеру ошибки.

Функция потока должна иметь заголовок вида:

```
void * func_name(void * arg)
```

Имя функции, естественно, может быть любым. Аргумент arg – это тот самый указатель, который передается в последнем параметре функции pthread_create(). Функция потока может вернуть значение, которое затем будет проанализировано заинтересованным потоком, но это не обязательно. Завершение функции потока происходит, если (а) функция потока вызвала функцию pthread_exit(3); (б) функция потока достигла точки выхода; (в) поток был досрочно завершен другим потоком. Функция pthread_exit() представляет собой потоковый аналог функции _exit(). Аргумент функции pthread_exit(), значение типа void *, становится возвращаемым значением функции потока. Как (и кому?) функция потока может вернуть значение, если она не вызывается из программы явным образом? Для того, чтобы получить значение, возвращенное функцией потока, нужно воспользоваться функцией pthread_join(3). У этой функции два параметра. Первый параметр, pthread_join(), – это идентификатор потока, второй параметр имеет тип «указатель на нетипизированный указатель». В этом параметре функция pthread_join() возвращает значение, возвращенное функцией потока. Конечно, в многопоточном приложении есть и более простые способы организовать передачу данных между потоками. Основная задача функции pthread_join() заключается, однако, в синхронизации потоков. Вызов функции pthread_join() приостанавливает выполнение вызвавшего ее потока до тех пор, пока поток, чей идентификатор передан функции в качестве аргумента, не завершит свою работу. Если в момент вызова pthread_join() ожидаемый поток уже завершился, функция вернет управление немедленно. Функцию pthread_join() можно рассматривать как эквивалент waitpid(2) для потоков. Попытка выполнить более одного вызова pthread_join() (из разных потоков) для одного и того же потока приведет к ошибке.

Посмотрим, как все это работает на практике. Ниже приводится фрагмент листинга программы *threads*, полный текст которой вы найдете на прилагаемом диске в файле **threads.c**:

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <pthread.h>
void * thread_func(void *arg)
{
  int i;
  int loc_id = * (int *) arg;
  for (i = 0; i < 4; i++) {
    printf("Thread %i is running\n", loc_id);
    sleep(1);
  }
}
int main(int argc, char * argv[])
{
  int id1, id2, result;
  pthread_t thread1, thread2;
  id1 = 1;
  result = pthread_create(&thread1, NULL, thread_func, &id1);
  if (result != 0) {
    perror("Creating the first thread");
    return EXIT_FAILURE;
  }
  id2 = 2;
  result = pthread_create(&thread2, NULL, thread_func, &id2);
  if (result != 0) {
    perror("Creating the first thread");
    return EXIT_FAILURE;
  }
  result = pthread_join(thread1, NULL);
  if (result != 0) {
    perror("Joining the first thread");
    return EXIT_FAILURE;
  }
  result = pthread_join(thread2, NULL);
  if (result != 0) {
    perror("Joining the first thread");
    return EXIT_FAILURE;
  }
  printf("Done\n");
  return EXIT_SUCCESS;
}
```



Рассмотрим сначала функцию thread_func(). Как вы, конечно, догадались, это и есть функция потока. Наша функция потока очень проста. В качестве аргумента ей передается указатель на переменную типа int, в которой содержится номер потока. Функция потока распечатывает этот номер несколько раз с интервалом в одну секунду и завершает свою работу. В функции main() вы видите две переменных типа pthread_t. Мы собираемся создать два потока, и у каждого из них должен быть свой идентификатор. Вы также видите две переменные типа int, id1 и id2, которые используются для передачи функциям потоков их номеров. Сами потоки создаются с помощью функции pthread_create().В этом примере мы не модифицируем атрибуты потоков, поэтому во втором параметре в обоих случаях передаем **NULL**. Вызывая pthread_create() дважды, мы оба раза передаем в качестве третьего параметра адрес функции thread_func, в результате чего два созданных потока будут выполнять одну и ту же функцию. Функция, вызываемая из нескольких потоков одновременно, должна обладать свойством реентерабельности (этим же свойством должны обладать функции, допускающие рекурсию). Реентерабельная функция – это функция, которая может быть вызвана повторно, в то время, когда она уже выполняется (отсюда и происходит ее название). Реентерабельные функции используют локальные переменные (и локально выделенную память) в тех случаях, когда их не-реентерабельные аналоги могут воспользоваться глобальными переменными.

Мы вызываем последовательно две функции pthread_join() для того, чтобы дождаться завершения обоих потоков. Если мы хотим дождаться завершения всех потоков, порядок вызова функций pthread_join() для разных потоков, очевидно, не имеет значения.

Для того, чтобы скомпилировать программу **threads.c**, необходимо дать следующую команду:

```
gcc threads.c -D_REENTERANT -I/usr/include/nptl -L/usr/lib/nptl --
lpthread -o threads
```

Команда компиляции включает макрос _REENTERANT. Этот макрос указывает, что вместо обычных функций стандартной библиотеки к программе должны быть подключены их реентерабельные аналоги. Реентерабельный вариант библиотеки *glibc* написан таким образом, что вы, скорее всего, вообще не обнаружите никаких различий в работе с реентерабельными функциями по сравнению с их обычными аналогами. Мы указываем компилятору путь для поиска заголовочных файлов и путь для поиска библиотек **/usr/include/nptl** и **/usr/lib/nptl** соответственно. Наконец, мы указываем компоновщику, что программа должна быть связана с библиотекой libpthread, которая содержит все специальные функции, необходимые для работы с потоками.

У вас, возможно, возникает вопрос, зачем мы использовали две разные переменные, id1 и id2, для передачи значений двум потокам? Почему нельзя использовать одну переменную, скажем id, для обоих потоков? Рассмотрим такой фрагмент кода:

```
id = 1;
pthread_create(&thread1, NULL, thread_func, &id);
id = 2;
pthread_create(&thread2, NULL, thread_func, &id);
```

Конечно, в этом случае оба потока получат указатель на одну и ту же переменную, но ведь значение этой переменной нужно каждому потоку только в самом начале его работы. После того, как поток присвоит это значение своей локальной переменной loc_id, ничто не мешает нам использовать ту же переменную id для другого потока. Все это верно, но проблема заключается в том, что мы не знаем, когда первый поток начнет свою работу. То, что функция pthread_create() вернула управление, не гарантирует нам, что поток уже выполняется. Вполне может случиться так, что первый поток будет запущен уже после того, как переменной id будет присвоено значение **2**. Тогда оба потока получат одно и то же значение id. Впрочем, мы можем использовать одну и ту же переменную для передачи данных функциям потока, если воспользуемся средствами синхронизации – им будет посвящена следующая статья.

Досрочное завершение потока

Функции потоков можно рассматривать как вспомогательные программы, находящиеся под управлением функции main(). Точно так же, как при управлении процессами, иногда у программы возникает необходимость досрочно завершить один из потоков. Для этого можно воспользоваться функцией pthread_cancel(3). Единственным аргументом этой функции является идентификатор потока. Функция pthread_cancel() возвращает 0 в случае успеха и ненулевое значение в случае ошибки. Несмотря на то, что pthread_cancel() может завершить поток досрочно, ее нельзя назвать средством принудительного завершения потоков. Дело в том, что поток может не только самостоятельно выбрать порядок завершения в ответ на вызов pthread_cancel(), но и вовсе игнорировать этот вызов. Вызов функции pthread_cancel() следует рассматривать как запрос на выполнение досрочного завершения потока. Функция pthread_setcancelstate(3) определяет, будет ли поток реагировать на обращение к нему с помощью pthread_cancel(), или не будет. У функции pthread_setcancelstate() два параметра, параметр state типа int и параметр oldstate типа «указатель на int». В первом параметре передается новое значение, указывающее, как поток должен реагировать на запрос pthread_cancel(), а в переменную, чей адрес был передан во втором параметре, функция записывает прежнее значение. Если прежнее значение вас не интересует,

во втором параметре можно передать **NULL**. Чаще всего функция pthread_setcancelstate() используется для временного запрета завершения потока. Допустим, мы программируем поток, и знаем, что при определенных условиях программа может потребовать его досрочного завершения. Но в нашем потоке есть участок кода, во время выполнения которого завершать поток крайне нежелательно. Мы можем оградить этот участок кода от досрочного завершения с помощью пары вызовов pthread_setcancelstate():

```
pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, NULL);
... //Здесь поток завершать нельзя
pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL);
```

Первый вызов pthread_setcancelstate() запрещает досрочное завершение потока, второй – разрешает. Если запрос на досрочное завершение потока поступит в тот момент, когда поток игнорирует эти запросы, выполнение запроса будет отложено до тех пор, пока функция pthread_setcancelstate() не будет вызвана с аргументом PTHREAD_CANCEL_ENABLE. Что именно произойдет дальше, зависит от более тонких настроек потока. Рассмотрим пример программы (вы найдете ее на диске в файле **canceltest.c**)

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
int i = 0;
void * thread_func(void *arg)
{
  pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, NULL);
  for (i=0; i < 4; i++) {
     sleep(1);
     printf("I'm still running!\n");
  }
  pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL);
  pthread_testcancel();
  printf("YOU WILL NOT STOP ME!!!\n");
}
int main(int argc, char * argv[])
{
  pthread_t thread;
  pthread_create(&thread, NULL, thread_func, NULL);
  while (i < 1) sleep(1);
  pthread_cancel(thread);
  printf("Requested to cancel the thread\n");
  pthread_join(thread, NULL);
  printf("The thread is stopped.\n");
  return EXIT_SUCCESS;
}
```

В самом начале функции потока thread_func() мы запрещаем досрочное завершение потока, затем выводим четыре тестовых сообщения с интервалом в одну секунду, после чего разрешаем досрочное завершение. Далее, с помощью функции pthread_testcancel(), мы создаем точку отмены (*cancellation point*) нашего потока. Если досрочное завершение потока было затребовано, в этот момент поток должен завершиться. Затем мы выводим еще одно диагностическое сообщение, которое пользователь не должен видеть, если программа работает правильно.

В главной функции программы мы создаем поток, затем дожидаемся, пока значение глобальной переменной i станет больше нуля (это гарантирует нам, что поток уже запретил досрочное завершение) и вызываем функцию pthread_cancel(). После этого мы переходим к ожиданию завершения потока с помощью pthread_join(). Если вы скомпилируете и запустите программу, то увидите, что поток распечатает четыре тестовых сообщения *I'm still running*! (после первого сообщения главная функция программы выдаст запрос на завершение потока). Поскольку поток завершится досрочно, последнего тестового сообщения вы не увидите.

Интересна роль функции pthread_testcancel(). Как уже отмечалось, эта функция создает точку отмены потока. Зачем нужны особые точки отмены? Дело в том, что даже если досрочное завершение разрешено, поток, получивший запрос на досрочное завершение, может остановиться не сразу. Если поток находится в режиме отложенного досрочного завершения (именно этот режим установлен по умолчанию), он выполнит запрос на досрочное завершение, только достигнув одной из точек отмены. В соответствии со стандартом POSIX, точками отмены являются вызовы многих «обычных» функций, например open(), pause() и write(). Про функцию printf() в документации сказано, что она может быть точкой отмены, но в Linux при попытке остановиться на printf() происходит нечто странное – поток завершается, но pthread_join() не возвращает управления. Поэтому мы создаем явную точку отмены с помощью вызова pthread_testcancel(). Впрочем, мы можем выполнить досрочное завершение потока, не дожидаясь точек останова. Для этого необходимо перевести поток в режим немедленного завершения, что делается с помощью вызова

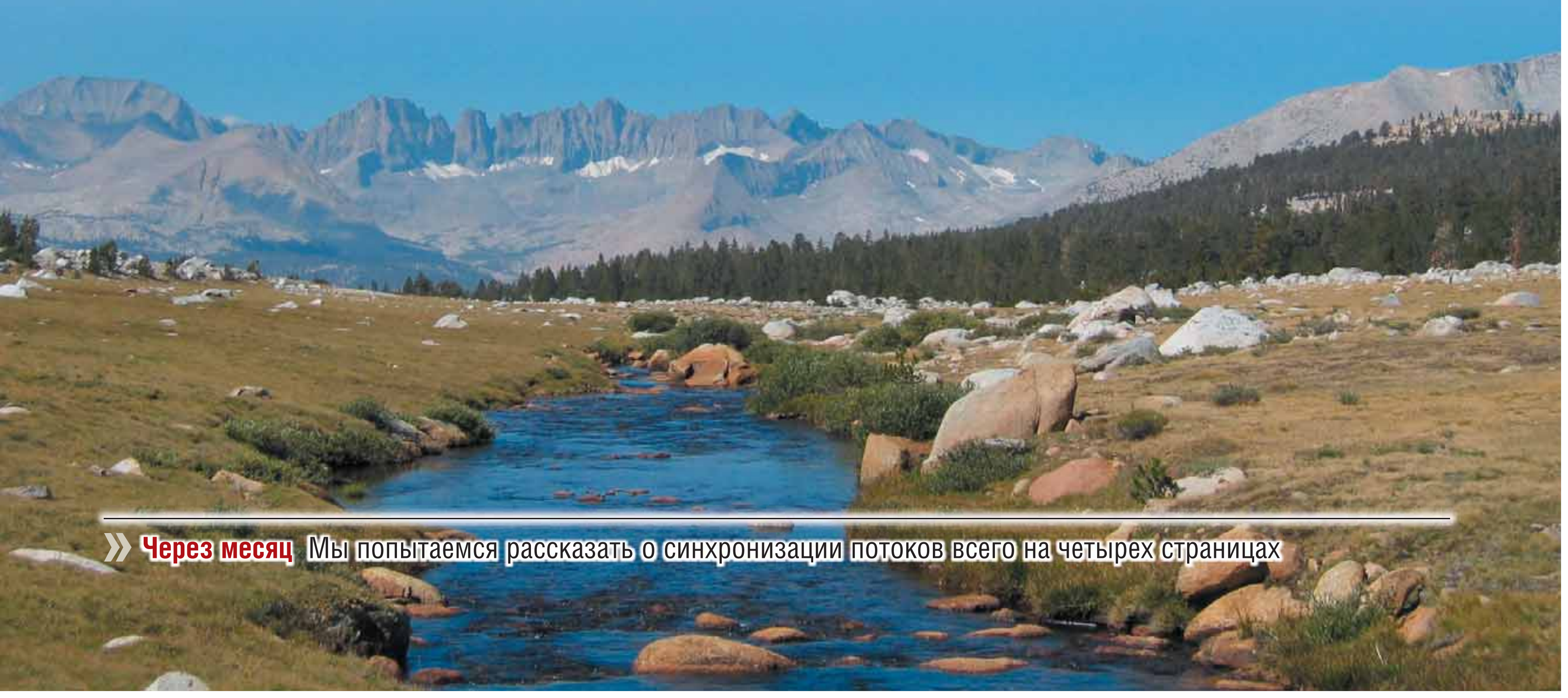
```
pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS, NULL);
```

В этом случае беспокоиться о точках останова уже не нужно. Вызов

```
pthread_setcanceltype(PTHREAD_CANCEL_DEFERRED, NULL);
```

снова переводит поток в режим отложенного досрочного завершения.

Тема потоков практически неисчерпаема (простите за каламбур), но мы посвятим потокам еще лишь только одну статью, в которой рассмотрим вопросы синхронизации и атрибуты потоков. LXF



» **Через месяц** Мы попытаемся рассказать о синхронизации потоков всего на четырех страницах



ЧАСТЬ 3: Даже самой замечательной программе надо откуда-то черпать данные для своей работы. Данные, как известно, хранятся в файлах. Тему продолжает **Антон Черноусов**.



В предыдущей статье из цикла, посвященного программированию на Java, были рассмотрены вопросы организации простых вычислений, ветвлений, циклов, а также генерации и обработки исключений.

В течение третьего урока мы поговорим о работе с файлами, о протоколировании работы программы и коснемся методов работы с XML-данными.

Файлы – потоки

Сказочное королевство под руководством царевны Несмеяны (т.к. супруг практически всегда отсутствовал), благодаря талантам и приобретенным навыкам, стало разрастаться, и результаты полюдья простонапросто перестали помещаться в семейный чулан. Чтобы накапливать и хранить богатства, потребовались дополнительные помещения, роль которых для нас привычно играют файлы.

Отношение к файлам в Java достаточно непростое: если рассматривать файл как устройство для ввода/вывода информации – с этой точки зрения он подобен блоку памяти или экрану, интерфейс доступа к которому унифицирован: это поток. Но несмотря на унифицированный интерфейс, существует большое количество классов сходной функциональности, в которых легко запутаться.

Поток можно представить в виде ленточного конвейера с последовательным размещением или извлечением данных, при использовании которого задача программиста сводиться к осуществлению операций «поместить/читать» данные, а остальные детали реализации скрыты от него скрыты.

Чтение данных

Разнообразие классов для работы с файлами позволяет выбрать для себя ту связку, которая больше нравится. Лично я для чтения данных использую BufferedReader, InputStreamReader и FileInputStream. Собственно взаимодействие этих классов для программиста заканчивается в момент создания экземпляра BufferedReader, что делается следующим образом:

```
BufferedReader br = null;
 br = new BufferedReader(new InputStreamReader
(new FileInputStream(pathToFile), encoding));
```

В процессе создания участвуют строковые переменные, содержащие путь к файлу и кодировку, в которой производится считывание данных: pathToFile и encoding. В классах, работающих с файлами, считается обязательным создавать переменную для кодировки по умолчанию:

```
protected static String DEFAULT_ENCODING = "UTF-8";
```

Любой текстовый файл можно представить себе в виде набора строк, поэтому давайте реализуем метод для считывания содержимого файла в массив String[]. Далее представлен метод rippedCurrentFile(pathToFile, encoding) класса FileRipper, который извлекает данные из файла с помощью метода readLine() экземпляра класса BufferedReader:

```
protected boolean rippedCurrentFile(String pathToFile, String encoding) {
 // connecting to file
 BufferedReader br = null;
 try {
  br = new BufferedReader(new InputStreamReader(new FileInputStream(
pathToFile),
encoding));
 } catch (UnsupportedEncodingException e) {
  this.error = FILE_ERROR_UNSUPPORTED_ENCODING; return false;
 } catch (FileNotFoundException e) {
  this.error = FILE_ERROR_NO_FILE; return false;
 }
 // ripping the file
 String str = null;
 ArrayList allStrings = new ArrayList();
 try {
  while (!(str = br.readLine()).equals(null)) { allStrings.add(str); }
 } catch (IOException e) {
  this.error = FILE_ERROR_IO_READ; return false;
 } catch (NullPointerException e) {
  this.error = FILE_ERROR_END_OF_FILE;
 }
 // free the resources
 try {
  br.close();
 } catch (IOException e) {
```

Месяц назад Мы познакомились с простейшими типами данных и операциями над ними.

данных

```
this.error = FILE_ERROR_IO_CLOSE; return false;
}
this.allStrings = (String[]) allStrings.toArray(new String[0]); return true;
}
}
```

Полный код примера, в том числе код класса ConsoleToFileRipper, применяющий экземпляр класса FileRipper для извлечения данных из файла, можно найти на диске в каталоге **examples 1**.

Запись данных

Процесс записи данных в файл хоть и отличается от чтения, но тоже достаточно похож на организацию конвейера. Для записи я обычно использую связку BufferedWriter, OutputStreamWriter, FileOutputStream.

```
BufferedWriter out;
out = new BufferedWriter(new OutputStreamWriter
(new FileOutputStream(pathToFile), encoding));
```

Для освобождения ресурсов, которые используют экземпляры классов BufferedReader и BufferedWriter, необходимо вызвать метод close().

Далее приведу простой пример метода, который записывает строковый массив в файл (полный код метода расположен на диске в директории **examples 2**):

```
public boolean createCurrentFile(String pathToFile, String[] allStrings,
String encoding)
{
try {
BufferedWriter out;
out = new BufferedWriter(new OutputStreamWriter
(new FileOutputStream(pathToFile), encoding));
for (int i = 0; i < allStrings.length; i++) {
out.write(allStrings[i]); out.write('\n');
}
out.close();
} catch (IOException e) {
e.printStackTrace(); return false;
}
return true;
}
```

Свободный доступ

В представленных ранее примерах доступ к данным осуществляется последовательно, что не всегда удобно (хотя в большинстве случаев именно такой доступ и используется). Для осуществления чтения и записи данных из файла в произвольном порядке существует специальный класс RandomAccessFile, экземпляр которого создается следующим образом:

```
RandomAccessFile raf = new RandomAccessFile(pathToFile, mode);
```

При этом pathToFile – путь до файла, а mode – режим работы. mode может принимать значения: r (только чтение), rw (чтение-запись), rws (чтение-запись с синхронным сохранением содержимого и метаданных), rwd (чтение-запись с синхронным сохранением содержимого файла). К сожалению, кодировку указать нельзя. Огромным преимуществом подхода является то, что с помощью метода getFilePointer() можно узнать текущее месторасположение указателя, а с помощью метода seek() можно передвинуть указатель в необходимое место в файле. Я предпочитаю не использовать данный класс – считайте это личным предубеждением.

Протоколирование работы программы

Для контроля и анализа работы приложения существуют методы протоколирования. Популярным инструментом для этих целей в мире Java является библиотека Log4j, которая разрабатывается в Apache Software Foundation. Текущую версию можно загрузить с http://logging.apache.org/.

Для использования библиотеки необходимо создать конфигурационный файл, который описывает, что, куда и как нужно протоколировать. Log4j имеет три базовые составляющие: logger, appender и layout.

layout – это элементы, определяющие вид и содержание записей. Изначально имеется несколько заранее созданных layout-ов, а в случае необходимости можно создать свой собственный.

Appender – это элемент, определяющий местоположение протокола, с его помощью задается тип протоколирования:

- файловое протоколирование (FileAppender);
- консольное протоколирование (ConsoleAppender);
- протоколирование в базы данных (JDBCAppender);
- протоколирование на SMTP-сервера (SMTPAppender) и др.

Logger – это элемент, который обеспечивает протоколирование какого-либо события. Если обратиться к ранее приведенной аналогии ленточного конвейера, logger – это и есть тот самый конвейер, вызывая методы которого, мы формируем протокол работы программы. Элемент logger предусматривает следующие уровни протоколирования: DEBUG, INFO, WARN, ERROR, FATAL; уровням соответствуют методы класса org.apache.log4j.Logger: debug; info; warn; error; fatal.

Ниже представлен пример записей для конфигурационного файла нашего приложения, которые нужно сохранить в файл с названием **log.properties** (название файла может быть любым).

```
log4j.logger.simple=DEBUG, nameLogAppender
log4j.appender.nameLogAppender=org.apache.log4j.FileAppender
log4j.appender.nameLogAppender.File=nameLogFile.log
log4j.appender.nameLogAppender.layout=org.apache.log4j.SimpleLayout
```

Первая строка указывает используемый уровень logger (DEBUG) и appender (nameLogAppender). Далее идут настройки appender: указание типа – FileAppender. В третьей строке указываем путь до файла журнала, а в последней – формат записи.

Использовать экземпляр класса Logger можно примерно так (пример протоколирования приведен на диске в каталоге **examples 3**):

```
File propertiesFile = new File("log.properties");
PropertyConfigurator.configure(propertiesFile.toString());
Logger logger = Logger.getLogger("simple");
logger.info("the program has started");
```

Документы XML

Проектируя и создавая ПО, невозможно не столкнутся с миром XML (Extensible Markup Language). XML был создан в недрах World Wide Web Consortium (W3C) для преодоления ограничений языка HTML. Можно сказать, что HTML – один из самых успешных языков, область его использования с каждым годом растет (в основном в объемах). Не смотря на это, почему же W3C создал XML, и зачем вам использовать этот язык? В чем ограниченность HTML? Ответ на эти вопросы один: »



» XML был создан для обеспечения взаимодействия разнородных систем.

HTML, как и любой другой текстовый язык, не позволяет перенести смысл тех данных, которые он хранит. XML был разработан для решения этой задачи с прицелом на Web, но получился таким удачным, что его стали использовать практически везде. Суть XML в том, что он хранит семантический смысл данных, поэтому выполнив анализ такого XML-документа, система может «понять» полученные данные. В Интернете существует большое количество информации, посвященной XML (например, www.ibm.com/developerworks/xml), поэтому не будем подробно останавливаться на его преимуществах, а сразу приступим к использованию.

```
<?xml version="1.0" ?>
<line name="firstLine">
 <point id="1" theX="1" theY="1"/>
 <point id="2" theX="2" theY="2"/>
</line>
```

Выше приведен простой пример XML-документа, содержащего корневой элемент line, который, в свою очередь, содержит два узла с тремя атрибутами (id, theX, theY) каждый.

Создание XML-документа

Говоря про создание XML-документа, я подразумеваю создание дерева XML-документа в памяти системы, то есть объекта DOM (Document Object Model). DOM была создана W3C, и это – официальная Рекомендация консорциума. В противовес DOM существует SAX (Simple API for XML). С моделью SAX можно ознакомиться более под-робно в http://www-128.ibm.com/developerworks/ru/views/xml/libraryview.jsp. Основная разница между методами заключается в том, что DOM обеспечивает виртуальное представление XML-файла в памяти системы, в то время как SAX – это событийная модель обработки, в которой в момент встречи определенного элемента вызывается соответствующее событие.

Для создания «отображения» XML-файла в памяти системы необходимо воспользоваться классом Document, экземпляр которого можно получить следующим образом:

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.newDocument();
```

Создание корневого узла XML документа можно выполняется так:

```
Element root = doc.createElement("line");
root.setAttribute("name", "firstLine");
```

С помощью метода createElement(name) производится создание элементов, в то время как создание и установка значений атрибутов элементов производится с помощью метода setAttribute(name, value), где name – имя атрибута, а value – его значение. С помощью метода appendChild можно добавить узел в элемент или произвести запись элемента в документ XML, например, так:

```
root.appendChild(item);
doc.appendChild(root);
```

Более детально процесс создания XML документа на примере класса Line изложен в директории **examples 4**. При вызове метода createDoc() этого класса создается экземпляр класса Document, идентичный приведенному ранее коду XML.

Сохранение XML-документа

Для сохранения созданного в памяти документа предлагаю воспользоваться методом saveXML(docToSave, pathToFile, charSet). В процессе сохранения участвует уже знакомый нам OutputStreamWriter, а также экземпляр класса Transformer, который обеспечивает преобразование объекта DOMSource в выходной поток.

```
public void saveXML(Document docToSave, String pathToFile, String
charSet) {
try {
 Writer target = new OutputStreamWriter(
new FileOutputStream(pathToFile), charSet);
 Source source = new DOMSource(docToSave);
 StreamResult dest = new StreamResult(target);
 Transformer t = TransformerFactory.newInstance().newTransformer();
 t.setOutputProperty(OutputKeys.ENCODING, charSet);
 t.setOutputProperty(OutputKeys.INDENT, "yes");
 t.transform(source, dest);
 target.flush();
 target.close();
} catch (Exception ex) {
 ex.printStackTrace();
}
}
```

Реализация этого метода представлена на диске в директории **examples 5**.

Загрузка XML-документа

Подобно сохранению XLM-документа, загрузка также достаточно просто выполняется с помощью экземпляра класса FileInputStream. Используя DocumentBuilderFactory, сгенерируем объект класса Document следующим образом:

```
 FileInputStream fis = new FileInputStream(pathToFile);
 Document doc =
 DocumentBuilderFactory.newInstance().newDocumentBuilder().
parse(fis);
```

Фактически, создание документа происходит вследствие выполнения метода parse. Для детально рассмотрения процесса загрузки XML документа обратитесь к директории **examples 6**, расположенной на диске.

Извлечение данных из XML-документа

Итак, что же делать с Document? Работать, естественно! Прежде чем приступить к обработке XML-документа, хочу обратить внимание на одну неприятную особенность: почти всегда XML документ содержит пустые узлы или символы перевода каретки (неизбежное зло форматирования). К сожалению, такая особенность существенно затрудняет процесс обработки XML-документа. Поэтому следует всегда проводить нормализацию с помощью определенного в интерфейсе метода normalize(), или, если реализация Document не имеет такой возможности или нормализация выполняется некорректно – воспользуйтесь методом normalizeDocument класса FileXMLReader (вы найдете его все в той же директории **examples 6**). Вызвать данный метод можно следующим образом:

```
normalizeDocument(doc.getDocumentElement());
```

Изложенный выше материал получился несколько другого формата, нежели предыдущие статьи. Статья содержит небольшие примеры и отсылает на набор сознательно упрощенных готовых решений, благодаря которым вы сможете работать с файлами, производить протоколирование деятельности вашего приложения, начать работать с данными в формате XML. Цель приведенных примеров – обеспечить вас необходимым минимумом информации и дать направление для поиска ответов на Ваши вопросы.

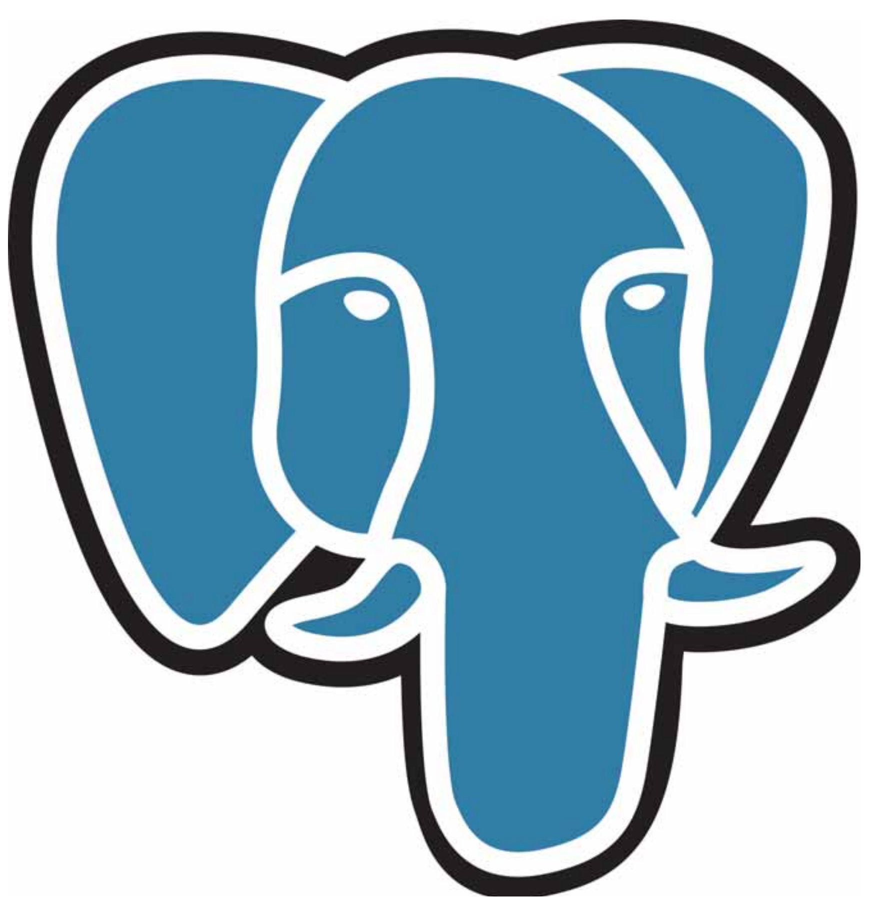
» **Через месяц** Мы научим Несмеяну работать с файлами и вести дневник.

Работа

ЧАСТЬ 2: Проблема компьютеров в том, что они делают то, что вы сказали, а не то, что подумали. Поэтому запустить PostgreSQL недостаточно – нужно еще суметь договориться с ним на понятном языке. **Евгений Балдин** открывает русско-машинный словарь...

Это прибор, – сказал Корнеев безнадежно. – С ним работают...

«Понедельник начинается в субботу»



В прошлой части мы узнали, как создать базу данных и запустить *postmaster*. Теперь дело за малым: надо научиться сохранять данные и «доступаться» до них. Для этого следует договориться с *postmaster* – благо его «родной» язык довольно высокоуровневый.

Как и в предыдущей части, все рассматривается с точки зрения дистрибутива Debian (Sarge). При прочтении следует иметь это в виду.

SQL

В качестве языка общения с реляционными базами данных в подавляющем большинстве случаев используется *SQL*. Изначально эти три буквы были сокращением фразы Structured Query Language (язык структурированных запросов). Сейчас, когда язык стал стандартом, *SQL* уже не является аббревиатурой – это обычное название, которое произносится как «эс-кью-эл». Несмотря на это, даже англоязычные специалисты по прежнему часто называют *SQL* «сиквел». По-русски также часто говорят «эс-ку-эль».

У этого языка есть недостатки, приводящие к тому, что в реальности *SQL* дополняется различными расширениями. Кстати, сам Кодд, «отец» реляционных баз данных, считал *SQL* неудачной реализацией его теории. Но на сегодня это мощный открытый промышленный стандарт, который позволяет решать множество типовых задач по созданию, модификации и управлению данными – он есть здесь и сейчас.

За время своего существования *SQL* претерпел несколько ревизий. Основные вехи в истории стандарта перечислены в таблице.

Степень соответствия *PostgreSQL* стандарту SQL:2003 подробно рассмотрена в Приложении D (*Appendix D. SQL Conformance*) стандартной документации.

Там же есть и простейший учебник, и исчерпывающий справочник по *SQL*. Существует море литературы, в которой подробно и не очень рассказывается, что же это за «зверь такой» – *SQL*. Необходимый для «вхождения в технологию» минимум настолько прост, что основы изучаются в пределах одного дня вдумчивого чтения учебника.

Для того, чтобы куда-то сохранить данные, необходимо создать «хранилище» – таблицу/таблицы:

```
CREATE TABLE fiodata (id int,fio text)
CREATE TABLE phonedata (id int,number text)
```

Теперь можно добавлять данные:

```
INSERT INTO fiodata VALUES (1,'Иванов И.П.')
INSERT INTO phonedata VALUES (1,'555-32-23')
```

Год	Ревизия	Нововведения
1986	SQL-86, SQL-87	Первая версия стандарта ANSI. Принят ISO в 1987 году. Стандартизация синтаксиса.
1989	SQL-89	Стандартизован механизм ссылочной целостности.
1992	SQL-92 (SQL-2)	Множество нововведений. В отличие от предыдущих версий, где стандарт просто сертифицировал уже имеющиеся на рынке реляционных БД возможности, были заложены основы для развития языка. Введены три уровня соответствия стандарту: Entry (начальный), Intermediate (промежуточный), Full (полный). Мало какая из баз данных поддерживает SQL-92 лучше, чем Entry.
1999	SQL:1999 (SQL-3)	Добавлены регулярные выражения, рекурсивные запросы, триггеры. Определена интеграция с объектно-ориентированным подходом. Вместо трех уровней соответствия введен набор свойств (features).
2003	SQL:2003	Стандартизованы XML-зависимые нововведения, интервальные функции (window functions), стандартные последовательности и столбцы с автоматически генерируемыми значениями.

с базой

и так далее. Мы создали две обычных таблицы «без наворотов»: в одной хранятся имена, а в другой – телефоны. Сопоставление телефонов именам происходит через поля id. Почему так? На одно имя может быть заведено несколько телефонов, а на одном телефоне может «сидеть» несколько человек.

Теперь надо извлечь данные, и в этом нам поможет оператор SELECT. Собственно говоря, пользователю, кроме этого оператора, больше ничего знать и не надо – все выборки делаются с его помощью. Выведем все имена и соответствующие им телефоны:

```
SELECT fio, number
     FROM fiodata,phonedata WHERE fiodata.id=phonedata.id
```

SQL, очевидно, заслуживает большего, чем это «микроввведение», и его, в любом случае, придется изучать тем, кто реально хочет заниматься базами данных – то есть от книжек никуда не денешься. А если подходить к делу серьезно, то кроме описания *SQL* следует изучить и основы реляционных баз данных того же К.Дж. Дейта (C.J. Date) – но это уже совсем другая история.

Командная строка

Когда набирается текст, а *SQL* – это именно текст, то лучше, чтобы ничего вокруг не отвлекало. Надежная, «толстая» и дешевая связь – вещь хорошая, только вот случается она не всегда, так что командная строка оказывается вне конкуренции.

psql

Вместе с пакетом *postgresql-client* поставляется утилита *psql* – интерактивная оболочка для «разговоров» с *PostgreSQL*. Она же – лучший инструмент для администрирования.

```
baldin@evgueni:~$ psql test
Добро пожаловать в psql 7.4.7 - Интерактивный Терминал PostgreSQL.

Наберите:  \copyright для условий распространения
           \h для подсказки по SQL командам
           \? для подсказки по внутренним slash-командам (\команда)
           \g или ";" для завершения и выполнения запроса
           \q для выхода

test=# select fio,number from fiodata,phonedata where fiodata.id=phonedata.id;
    fio     |     number
------------+----------------
 Иванов И.П. | 555-32-23
 Балдин Е.М. | 555-41-37
 Балдин Е.М. | (+7)5559323919
(записей: 3)

test=#
```

› **Окно *psql*.**

Пусть существует база данных test, в которой заведены таблицы fiodata и phonedata, описанные в предыдущем разделе. Подсоединимся к базе и что-нибудь «спросим» у нее:

```
> psql test
```

Добро пожаловать в *pSQL* 7.4.7 – Интерактивный Терминал *PostgreSQL*.

```
Наберите:  \copyright для условий распространения
           \h для подсказки по SQL командам
           \? для подсказки по внутренним slash-командам (\команда)
           \g или ";" для завершения и выполнения запроса
           \q для выхода

test=> SELECT fio, number
test->       FROM fiodata,phonedata WHERE fiodata.id=phonedata.id;
   fio      |     number
-------------+----------------
 Иванов И.П. | 555-32-23
 Балдин Е.М. | 555-41-37
 Балдин Е.М. | (+7)5559323919
(записей: 3)
```

Если хочется подсоединиться к серверу на другой машине, то нужно указать имя машины после ключа -h. Ключ -U позволяет указать имя пользователя.

psql передает *SQL*-команды на сервер. Обратите внимание, что для завершения *SQL*-команды используется точка с запятой – ;.

Как и всякая человеко-ориентированная оболочка, *pSQL* использует библиотеку *Readline*. Это означает наличие стандартных горячих emacs-подобных комбинаций символов для общепринятого редактирования ввода командной строки, в том числе и завершение *SQL*-команд по Tab. По клавише Tab завершаются не только *SQL*-команды, но и названия таблиц и имена колонок, если это возможно.

psql поддерживает историю команд, которая сохраняется в **.pSQL_history**. Это также особенность библиотеки Readline. Полезным является интерактивный поиск по истории команд, который вызывается с помощью комбинации C^r.

Кроме команд *SQL*, *psql* имеет набор собственных специальных команд. Все такие команды начинаются с обратной косой черты \. Число спецкоманд довольно обширно и полное их описание можно распечатать, выполнив команду man pSQL. Далее будет перечислены наиболее интересные из них:

- \q Закончить работу с p*SQL*. Выйти из оболочки.
- \? Вывести справку по имеющимся спец-командам.
- \h [*SQL*-команда] Вывести помощь по запрашиваемой *SQL*-команде в форме Бэкуса-Наура (Backus Naur Form). *SQL*-команда может состоять из нескольких слов. При исполнении \h без аргумента выводится список доступных *SQL*-команд.
- \! [shell-команда] Запустить командный интерпретатор и выполнить команду оболочки.
- \i «файл» Прочитать текстовый файл и выполнить имеющиеся в нем команды. Удобно для нетривиальных операций.

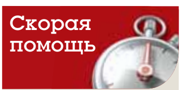
Имя файла с командами можно передать при запуске *psql* посредством ключа -f. В этом случае после чтения и исполнения всех команд *psql* автоматически прекращает работу.

- \o [«файл»] Сохранить результаты выполнения будущих команд в файл. Если аргумент отсутствует, то вывод переключается на терминал. *psql* имеет набор команд, которые позволяют сформировать вывод в удобном для пользователя виде.

Имя файла, в котором следует сохранить результаты, также можно передать при запуске *psql* с помощью ключика -o. Этот ключ удобно применять совместно с ключом -f.

»

Скорая помощь



После первого запуска *PgAccess* необходимо настроить шрифты: Database → Preferences → Look & Feel. Логика их выбора по умолчанию не совсем, с моей точки зрения, адекватна, с другой стороны – при желании это настраивается.

» \d [«регулярное выражение»] Вывести структуру объекта. Годится для таблицы (table), представления (view), индекса (indexes) или последовательности (sequences). Список объектов можно получить, добавив первую букву названия объекта t, v, i, s к команде \d.

В дополнение к вышесказанному, *psql* поддерживает простейший механизм присваивания значений собственным переменным и их интерполяции в *SQL*-запросах:

```
test=> \set proba 'phonedata'
test=> select * from :proba;
 id |     number
----+----------------
  2 | 555-41-37
  2 | (+7)5559323919
  1 | 555-32-23
(записей: 3)
```

Следует учитывать, что интерполяция переменных не работает, если переменная используется внутри *SQL*-строки. В любом случае, хоть какое-то подспорье.

gql-shell

Небольшая *psql*-подобная оболочка, написанная одним человеком. Разработка заморожена. Естественно, она не обладает всеми возможностями *psql*, зато может подсоединяться и «разговаривать» не только с *PostgreSQL*. Для подсоединения к базе данных используется библиотека *GQL* (Generic C++ SQL Library). Для работы с *PostgreSQL* необходимо установить драйвер:

```
> sudo apt-get install gql-shell
> sudo apt-get install libgql-driver-0.5-pg
> gql-shell pg:test
Welcome to gql-shell, the interactive SQL terminal.

Type:  \copyright for distribution terms
\h for help with SQL commands
\? for help on internal slash commands
\g or terminate with semicolon to execute query
\q to quit

test=>
```

dbishell

dbishell – интерактивная оболочка на основе Perl::DBI. Как и *gql-shell*, поддерживает не только *PostgreSQL*. *dbishell* представляет из себя скрипт на Perl и занимает при установке чуть больше 150 Кб.

```
> sudo apt-get install dbishell
> dbishell --driver Pg --dsn host=localhost\;database=test --user baldin
Password:
Using DBIShell::dr::Pg engine

dbi:Pg:host=localhost;database=test:baldin>quit/
```

Для завершения любой команды используется косая черта /.

GUI вам в помощь

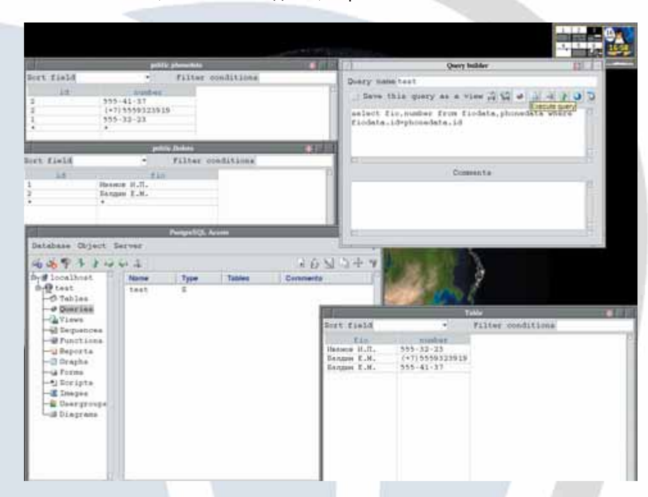
Следует признать, что программа с графическим пользовательским интерфейсом выглядит гораздо солиднее какой-то там командной строки. Об эффективности работы в случае необходимости показать, что занят важным делом, речь, естественно, не идет. Зачем один терминал, когда можно открыть кучу красивеньких окошечек с иконками? Конечно, *PostgreSQL* позволяет разговаривать с собой и через окна.

PgAccess

Когда обсуждается графический пользовательский интерфейс к *PostgreSQL*, тут же всплывает слово *PgAccess* (http://www.pgaccess.org/). *PgAccess* создан Константином Теодореску (Constantin Teodorescu) и имеет довольно длительную историю развития. На текущий момент разработка, похоже, заморожена. С другой стороны, «нет худа без добра»: новых версий тащить не надо – достаточно поставить то, что идет стандартно с вашим дистрибутивом:

```
> sudo apt-get install PgAccess
> PgAccess
```

Для того, чтобы подсоединиться к базе данных необходимо воспользоваться диалогом открытия соединения: Database → Open. По умолчанию предполагается, что *postmaster* запущен на этом же компьютере (Host: localhost), и он «слушает» порт номер 5432 – если при установке *PostgreSQL* ничего специально не делалось, то так оно и есть. Далее требуется указать базу данных, к которой надо подсоединиться, пользователя и, если необходимо, пароль.



› ***PgAccess* явно что-то умеет.**

PgAccess – это кросс-платформенный графический интерфейс к *PostgreSQL* написанный на чистом Tcl/Tk, и как следствие этого, работает везде, где этот инструментарий имеется (даже на «альтернативной» платформе). Размер дистрибутива по современным меркам крошечный: при установке все укладывается в 4 Мб.

В программе есть возможность создавать, редактировать и просматривать таблицы, запросы, представления, функции, пользователей, то есть довольно многое из того, что можно делать с помощью *SQL*. Плюс к этому можно создавать графические формы для ввода/просмотра данных, рисовать простые диаграммы и графики, просматривать картинки, сохраненные в базе данных. Так как программа написана на Tcl/Tk, то есть возможность писать свои скрипты, используя объекты, уже определенные в *PgAccess*.

Если хочется «сляпать» на скорую руку формочку, которую можно запустить фактически где угодно после минимальной доводки, то *PgAccess* вполне может подойти для этого дела. В самом *PgAccess* масса ограничений и недостатков, но так как программа относительно небольшая, то ее можно доделать «по месту».

Информацию о созданных формах, запросах и тому подобных объектах *PgAccess* сохраняет непосредственно в базе данных, в таблицах, начинающихся с префикса pga_. Так что то, что сделано кем-то одним, будет доступно и всем пользователям базы.

Наличие *PgAccess* на машине, с моей точки зрения, поощряет нездоровое желание что-то «сляпать», а не сделать по-человечески. Так что работать с этим предметом надо осторожно, и если нет необходимости, то лучше убрать его от греха подальше. По мне, так p*SQL* гораздо удобнее и эффективнее, а самое главное – пользователи наведываются в БД гораздо реже.

pgAdmin III

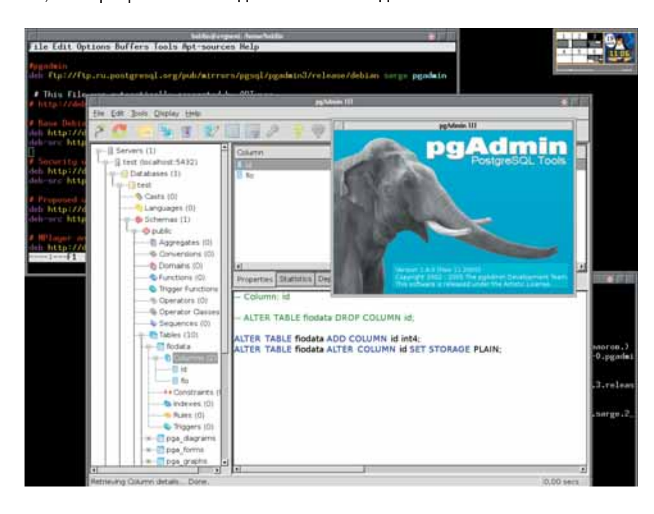
Программа порадует вас заявлением, что она наиболее популярная и функциональная платформа администрирования и разработки для *PostgreSQL*, а также своим отсутствием в дистрибутиве Debian (Sarge), посему установка начнется с ее выкачивания. К счастью, на сайте проекта http://www.pgadmin.org/ можно найти сборки под множество дистрибутивов. Есть и специальный репозитарий для Debian – в **/etc/apt/source.list** добавляется строка:

```
deb [MIRROR URL]/pgadmin3/release/debian sarge pgadmin
```

где вместо [MIRROR URL] подставляется одно из официальных зеркал *PostgreSQL*, например: **ftp://ftp.ru.PostgreSQL.org/pub/mirrors/pgSQL**, и производится установка программы:

```
> sudo apt-get update
> sudo apt-get install pgadmin3
> pgadmin3
```

При этом скачивается около 7,5 Мб. После запуска можно убедиться, что программа выглядит вполне солидно.



› ***pgAdmin* III подробно объясняет что надо «сказать» чтобы создать выбранный объект.**

Новое соединение создается через меню File → Add server. Требуется указать Address (localhost для локальной машины), сделать краткое описание соединения (Description), выбрать базу данных (Maintenance DB) и пользователя. После подсоединения доступны все объекты, которыми может управлять текущий пользователь.

PgAdmin III – это продукт для администрирования и управления базами данных под управлением *PostgreSQL* и его потомков. *PgAdmin III* содержит в себе графический интерфейс для управления данными, *SQL*-редактор с графическим представлением EXPLAIN, имеет инструменты для создания и редактирования таблиц, умеет управляться с системой репликации Slony-I и многое другое, что действительно упрощает администрирование. И все-таки *PgAdmin III* не для пользователя. Пока нет понимания того, что происходит, не следует уповать на картинки.

Изначально *pgAdmin* разрабатывался под Windows, но на сегодня этот продукт является многоплатформенным решением и работает также под Linux, Mac OSX, FreeBSD и *Solaris*. В качестве графической библиотеки используется *wxWidgets* (**http://www.wxwidgets.org**).

Русский перевод интерфейса, в принципе, существует, но на текущий момент не поддерживается. С другой стороны, это прежде всего инструмент администрирования и управления данными. В любом случае, разработчики приветствуют новые и обновленные переводы.

TOra

TOra возникла благодаря тому, что Генри Джонсон (Henrik Johnson) не смог запустить VMWare с Windows в далеком 2000 году. В то же время ему хотелось иметь графическую утилиту для администрирования Oracle, подобную той, которой пользовались его друзья, так и не отошедшие от «окон». *TOra* – это *toolkit for Oracle*. Так было, но на сегодня (в том числе и вследствие того, что *TOra* написана с использованием библиотеки *Qt3*), также можно работать и с *PostgreSQL*. Кроме *PostgreSQL*, дополнительно поддерживается *MySQL* и все, что работает через ODBC.

»

[1] Вследствие того, что PostgreSQL распространяется под лицензией BSD, имеется несколько коммерческих продуктов, основанных на его коде, например, EnterpriseDB, Pervasive Postgres и SRA PowerGres.0

PostgreSQL в лицах: Федор Геннадьевич Сигаев



Возраст: 33 года. Дипломированный физик. Профессиональный программист.
Место работы: «свободный художник».
Домашняя страничка: http://www.sigaev.ru/.

Федор Сигаев постоянно улучшает качество и скорость поиска в *PostgreSQL*. Он сделал значительный вклад в развитие GiST (Generalized Search Tree) и полнотекстовый поиск. Ему нравится развивать различные алгоритмы индексации данных.

Программированием Федор занялся вплотную, еще будучи студентом. Дипломная работа была посвящена построению математической модели межпланетного магнитного поля, и модель оказалась аналитически невычислимой, поэтому существенный объем диплома занимала программная часть. Окончание обучения совпало с «интересными временами», что привело к смене профессии, благо необходимые навыки уже были, и интерес к программированию наличествовал.

Евгений М. Балдин (ЕМБ): Меня интересуют причины, по которым вы включились в проект *PostgreSQL*. Как это случилось?

Федор Г. Сигаев (ФГС): *PostgreSQL* я занялся во время работы в Рамблере (http://www.rambler.ru). Нужен был онлайновый полнотекстовый поиск (сейчас это модуль *tsearch2*) для новостей, хранящихся в *PostgreSQL*. Так и пошло: сначала был создан модуль полнотекстового поиска, потом его индексная поддержка, а затем начали появляться улучшения уже в самом *PostgreSQL*. По мере того как я разбирался с исходными текстами, появлялись идеи, как его улучшить, какие типы данных для решения портальных задач необходимо добавить. Таким образом появились иерархический тип данных (ltree), поиск похожих слов (pg_trgm) и т.д.

ЕМБ: То, что вы заняты в таком проекте, не мешает вам заниматься основной работой?

ФГС: В настоящее время это и является моей основной работой. Ранее работодателям всегда удавалось объяснить/доказать необходимость моей работы над *PostgreSQL*. А сейчас заказчики находят нас специально именно для доработки или усовершенствования *PostgreSQL*.

ЕМБ: Собственно говоря, чем это интересно? Что такого притягательного в индексах?

ФГС: Всегда любил ковыряться поближе к основам. В программировании мне интересны алгоритмы, а не, условно говоря, пользовательские интерфейсы. А потроха базы данных – это алгоритмы и идеи.

В основу *PostgreSQL* заложена очень мощная идея – table-driven database. То есть, все операторы, типы, индексы и т.д. не вшиты в код, а доступны через системные таблицы. Например, для каждого типа есть функции преобразования из человеко-читаемого вида в машинный и обратно. Названия этих функций просто лежат в системной таблице, хранящей описания типов данных. Таким образом, *PostgreSQL* намного легче расширять дополнительными модулями без перекомпиляции самой базы данных. Быстрый результат приятен вдвойне.

Азбука SQL

Язык манипулирования данными

Группа операторов *SQL*, ответственных за добавление, удаление и модификацию данных, представляет из себя специализированный язык манипулирования данными (Data Manipulation Language).

Язык манипулирования данными включает следующие операторы:

- INSERT – позволяет добавить одну или несколько строк (rows) данных в уже существующую таблицу,
- UPDATE – позволяет изменить уже существующие данные,
- DELETE – позволяет удалить одну и более строк данных из таблицы,
- TRUNCATE – позволяет очистить одну или несколько таблиц от данных (очень опасная команда).

Для вставки данных следует воспользоваться командой:

```
INSERT INTO «имя таблицы»
        [(«список столбцов»)] VALUES («список
значений»)
```

Столбцы и значения в соответствующих списках разделяются запятыми. При отсутствии списка столбцов значения присваиваются в соответствии с порядком именования столбцов при создании таблицы.

В качестве значения команде INSERT можно передать **null**. Это эквивалентно тому, что соответствующее поле не инициализируется при вставке.

Для модификации данных следует использовать команду:

```
UPDATE «имя таблицы»
   SET «столбец»=«значение» [,…]
  [ WHERE «условное выражение» ]
```

Условные выражения могут объединяться по «и» (AND) или по «или» (OR). Логика в SQL – трехзначная. Кроме ожидаемых значений для условных выражений, таких как «истина» (**true**) и «ложь» (**false**), допустимо значение «не определено» (**unknown**).

Таблица 1: Таблица истинности для оператора OR.

OR	TRUE	FALSE	UNKNOWN
TRUE	true	true	true
FALSE	true	false	unknown
UNKNOWN	true	unknown	unknown

Таблица 2: Таблица истинности для оператора AND.

AND	TRUE	FALSE	UNKNOWN
TRUE	true	false	unknown
FALSE	false	false	false
UNKNOWN	unknown	false	unknown

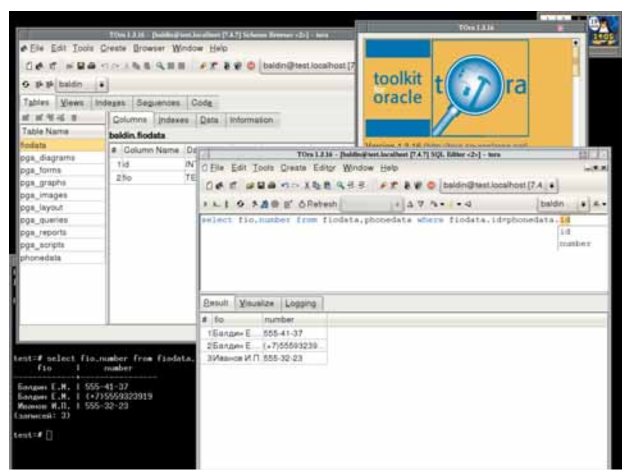
Для удаления данных используются команды:

```
DELETE FROM «имя таблицы»
  [ WHERE «условное выражение» ]
```

или

```
TRUNCATE TABLE  «имя таблицы» [, …]
```

Как это ни печально, но удалять гораздо проще, чем добавлять их. Не следует злоупотреблять этими операторами.



› *TOra* знает все об *SQL* и кое-что сверх того.»

Установка и запуск *TOra* просты:

```
> sudo apt-get install TOra
> TOra
```

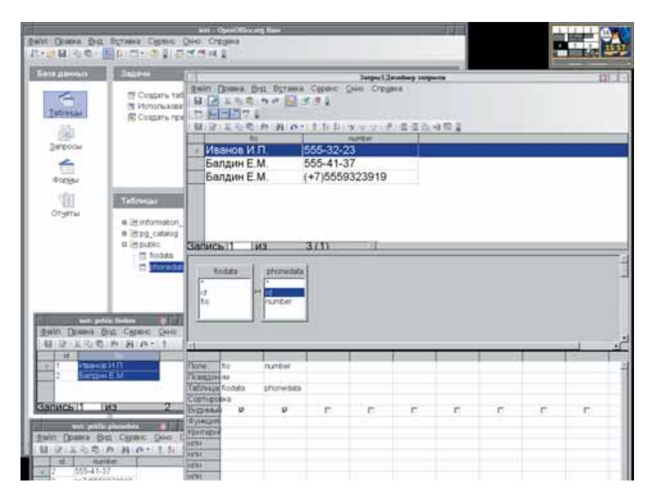
TOra предлагает диалог создания нового соединения сразу при старте. Требуется указать Connection provider (*PostgreSQL*), Username, Host (localhost), Port (5432) и DataBase.

Из-за прошлого программы, многие возможности *TOra* оказались привязаны к особенностям Oracle. В случае работы с *PostgreSQL*, *TOra* полезна прежде всего как браузер по *SQL*-объектам, *SQL*-терминал и изощренный *SQL*-редактор. Как и в случае pgAdmin III, *TOra* позволяет создавать и редактировать таблицы с помощью графических диалогов, но не владеет специфичными для *PostgreSQL* настройками.

TOra – это крепко «сбитый» программный продукт, который позволяет работать с разными реляционными *СУБД* в пределах одной программы.

OpenOffice.org и SDBC

OpenOffice.org – монстр, но ситуация на сегодня такая, что люди любят монстров, и ничего в обозримом будущем с этим не поделаешь.



› *OpenOffice* + *PostgreSQL*.»

Для прямого доступа из *OpenOffice* к *PostgreSQL* без промежуточного уровня в виде ODBC/JDBC драйверов разрабатывается драйвер *PostgreSQL-sdbc*. На сегодня в стандартной поставке *OpenOffice.org* этот пакет отсутствует.

Для установки необходимо скачать zip-архив этого драйвера с его домашней странички http://dba.OpenOffice.org/drivers/PostgreSQL/index.html и положить куда-нибудь у себя на диске не распаковывая (!).

Далее, запустив *OpenOffice*, следует открыть диалог управления пакетами: Сервис → Управление пакетами… и с помощью кнопки Добавить установить этот пакет. В моем случае после установки пришлось перезапустить *OpenOffice*.

Для подсоединения к уже существующей базе данных *PostgreSQL* следует открыть диалог Мастера базы данных:

Создать → Базу данных → Выбор базы данных, поставить галочку Подключиться к существующей базе данных и выбрать PostgreSQL. Далее при настройке соединения в следует ввести строчку вида:

```
dbname="имя БД" host="адрес сервера"
```

подставив вместо имя БД и адрес сервера имя предварительно созданной базы данных и адрес сервера, на котором «крутится» *postmaster*, например, dbname=test host=localhost. Далее, во вкладке Аутентификация пользователя необходимо ввести имя пользователя – затем можно протестировать соединение. Если тест прошел нормально, то можно продолжить и выполнить подключение.

Во время окончания действия мастера предлагается сохранить все, что проделано в ODB-файле (формат База данных OpenDocument). Затем это соединение можно будет выполнить простым открытием файла. Туда же сохраняется информация обо всех созданных формах, запросах и отчетах. Как конкретно создаются формы и отчеты – это совсем другая история? и относится она не к *PostgreSQL*, а к *OpenOffice.org* (см. LXF85).

При выборе таблиц легко видеть, что они в *PostgreSQL* разбиты на группы. Пользовательские таблицы по умолчанию находятся в группе public. В группах pg_catalog и informaion_schema представлена системная информация и статистика.

Что выбрать?

Естественно, мы рассмотрели далеко не все возможные программы общего назначения для работы с *PostgreSQL*, но даже из того, что рассмотрено, нельзя выбрать что-то одно. Каждая программа имеет свои особенности и преимущества. *psql* позволяет легко работать удаленно, *OpenOffice.org* удовлетворяет нашу любовь к монстрам, *PgAdmin III* содержит множество подсказок по делу, *PgAccess* удивляет своей интеграцией с TCL/Tk, а *TOra* – «просто красивая».

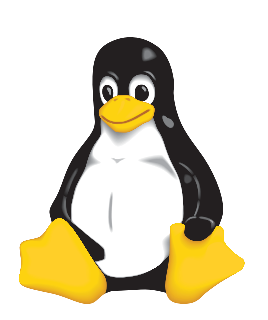
Я всегда выбирал *psql*, но это скорее всего связано со специфичностью решаемых мной задач. Я вполне могу представить себе ситуацию, когда наличие, например, *TOra* значительно облегчит жизнь. Ну и не следует забывать про данные, которые кто-то должен вводить. Если лень писать специальную программу (которую лучше все-таки написать), то *OpenOffice.org* поможет, особенно если вводить не вам. LXF

» **Через месяц** Мы узнаем, что умеет PostgreSQL.

Графика

ЧАСТЬ 3: Порой иллюстрация стоит тысячи слов – и сегодня Евгений Балдин расскажет, как добавить их в создаваемые вами документы LaTeX!




Q: Как быстро написать на LaTeX-е курсовую, в которой кроме текста есть и графики?
A: Сделать для начала графики.

Вопрос и краткий ответ

Вероятно, на текущий момент *TeX* лучше других программ вёрстки умеет разбивать абзацы на строки, то есть удачнее всех разливать порции «клея» между «боксами». Но подготовка графики выносится за рамки этого процесса. Почти…

С точки зрения *TeX*, картинка – это просто очень большой прямоугольник, который надо как-то разместить на странице. От пользователя нужны только размеры этого прямоугольника. Отображение же иллюстрации лежит на плечах драйверов. Самым востребованным форматом для представления графики в *LaTeX* до сих пор является Encapsulated PostScript.

Encapsulated PostScript

Уже больше двадцати лет прошло с тех пор, как никому не известная фирма Adobe Systems Inc. получила от компании Apple инвестиции на «обучение» лазерных принтеров молодому языку PostScript. Как следствие, этот платформенно-независимый язык с полностью открытой спецификацией стал безальтернативным стандартом. Даже сейчас PostScript фактически не имеет конкурентов в области допечатной подготовки. Поэтому почти все «уважающие себя» графические программы умеют экспортировать результаты своей деятельности в виде инструкций PostScript. Это особенно касается векторных графических редакторов, так как PostScript подразумевает векторную графику.

Encapsulated PostScript (EPS) – графический формат [скорее даже язык программирования, – прим. ред.]. Файлы в этом формате обычно имеют расширение **.eps**. По сути дела, это PostScript с некоторыми упрощениями и дополнительными договорённостями. Самая интересная с точки зрения *LaTeX* договорённость – это обязательное наличие в заголовке информации о размере картинки, которая передаётся вместе с комментарием:

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: dvips(k) 5.95b Copyright 2005 Radical Eye Software
%%Title: picture.dvi
%%BoundingBox: 127 464 430 667
%%DocumentFonts: SFRM1200 SFRM0800
%%EndComments
```

Первая строка комментария обычно содержит версию PostScript[1]. Вслед за комментарием BoundingBox идёт информация о размерах. Первые два числа соответствуют координатам левого нижнего угла картинки, а последние соответствуют координатам правого верхнего угла. Единицей измерения является «большой пункт» (bp=1/72 in), который примерно равен 0,351 мм. Для вёрстки текста указанной информации достаточно.

Чтобы из уже имеющегося одностраничного PostScript-файла сделать EPS, необходимо и, как правило, достаточно добавить BoundingBox. Для вычисления искомых размеров можно воспользоваться утилитой *ps2eps* из одноимённого пакета. Если же в стандартной поставке эта программа отсутствует, то можно напрямую воспользоваться программой *Ghostscript* – свободным программным интерпретатором PostScript:

```
> gs -q -dSAFER -dNOPAUSE -dBATCH -sDEVICE=bbox «имя файла»
```

Размеры выясняются с помощью указания специального драйвера bbox. Ключи -q, -dNOPAUSE и -dBATCH используются для подавления вывода ненужной информации и вопросов со стороны программы. Ключ -dSAFER гарантирует, что *Ghostscript* не будет производить никаких деструктивных действий[2].

Ещё одной особенностью формата EPS является возможность добавлять растровое изображение для предварительного просмотра. Это было сделано для ситуаций, когда программа не понимает PostScript, а что-то на месте «дырки» для картинки отобразить надо. Такое добавление идёт вразрез с принципиальной кросс-платформенностью PostScript и его следует, по возможности, избегать. Для операций с этим расширением, в том числе и для добавления/удаления растра, можно воспользоваться утилитой *epstool* из одноимённого пакета.

В конце рассказа про EPS хотелось бы упомянуть о замечательной утилите *pstoedit* из, естественно, одноимённого же пакета. Не все, но некоторые из более-

[1] Некоторые программы (не будем тыкать пальцем в драйвер для PostScript-принтеров одной очень распространённой операционной системы) добавляют перед комментарием бинарный мусор. Для полноценной работы с такими файлами этот мусор следует удалить.

Месяц назад Мы набирали математические формулы.

```
% Эмблемы \TeX{} и \METAFONT{}, созданные
%Дуайном Бибби, взяты со странички Д.Э.\,Кнута.

% Цветной пингвин взят из пакета \texttt{ps2pdf}
%от Rolf Niepraschk

\includegraphics[width=\textwidth]{title.1.eps}
```



› **Пример использования \includegraphics.**

Значение представляет из себя четыре числа, кодирующие положение левого нижнего и правого верхнего углов, например: [bb=127 464 430 667]. Вместо одного bb можно воспользоваться четвёркой ключей: [bbllx=127,bblly=464,bbrx=430,bbry=667], каждому из которых присваивается только одно значение.

Помимо перечисленных ключей, для модификации BoundingBox можно использовать viewport – четыре числа, описывающих границы BoundingBox, где в качестве центра координат выбирается левый нижний угол уже существующего описания и trim – четыре числа, представляющих собой отступы от левой, нижней, правой и верхней границ.

менее внятно созданных PostScript-файлов она ухитряется перевести в редактируемый векторный графический формат. Это упрощает правку файлов, которые не имеют исходных текстов.

Как сделать EPS из растра

Одним из важных вопросов является конвертация растровых форматов в EPS. Растр гораздо проще создавать, а кое-где, например, при снятии скриншотов, применим фактически только растр. Стандартные же средства преобразования, например, утилита *convert* из пакета *ImageMagick*, не всегда дают оптимальные результаты.

Возможным и вполне разумным решением является замена традиционной линейки: *LaTeX→dvips→[ps2pdf]* на *pdfLaTeX*, сразу «из коробки» поддерживающий растровые форматы PNG и JPEG, которые можно внедрять в формат PDF напрямую. Массового перехода на данную технологию пока не наблюдается, но заметное движение в эту сторону есть. У неё есть неоспоримые достоинства, но она не лишена недостатков. Рассказ о *pdfLaTeX* выходит за рамки этой статьи.

Вопрос о конвертации из JPEG можно решить с помощью простой утилиты *jpeg2ps*, которую можно найти в любом CTAN[3]-архиве в директории **nonfree/support/jpeg2ps**. Утилита не преобразовывает JPEG-файл, а просто добавляет правильный EPS-заголовок. Декомпрессия JPEG производится непосредственно интерпретатором PostScript[4]. К недостаткам утилиты можно отнести то, что в силу своей лицензии она не может распространяться со свободными дистрибутивами, а к достоинствам – отсутствие зависимостей.

Более комплексными решениями являются утилиты *sam2p* из одноимённого пакета и *bmeps*. Их также можно найти на CTAN в директориях **graphics/sam2p** и **support/bmeps**, соответственно. *sam2p* является своеобразным комбайном, который поддерживает множество растровых графических форматов, в то время как *bmeps* фокусируется на PNG и JPEG. Обе эти программы позволяют получить вполне приличную EPS-картинку для печати или просмотра на экране. В обоих случаях будет необходимо разобраться в ключах и настройках. С моей точки зрения, *bmeps* является более удобным решением, производящим достаточно маленькие[5] по размеру EPS-файлы, но и *sam2p* достаточно хорош.

Опять же на CTAN в директории **graphics/a2ping** можно взять довольно увесистый perl-скрипт **a2ping.pl**. Он является своеобразной надстройкой над *sam2p* и *Ghostscript*, что позволяет ему более-менее автоматически конвертировать из растра в PostScript и обратно.

Обзор внешних программ закончен. Внимание! Далее слово «пакет» будет относиться к пакетам *LaTeX*, а не пакетам дистрибутива GNU/Linux.

graphicx

Ответственным за создание «бокса» для размещения картинки является пакет *graphicx*[6], а точнее, команда \includegraphics.

В команде есть один обязательный параметр – вставляемая картинка. Необязательные параметры передаются с помощью пар «ключ=значение», разделяемых запятой. За подобный способ объявления параметров отвечает пакет *keyval*. Некоторые из поддерживаемых пакетом параметров перечислены ниже:

» bb – позволяет исправить BoundingBox прямо в коде, не меняя EPS-файл.

» clip – обрезает вставленную картинку по BoundingBox. Это необходимо делать в случае изменения границ для «выкусывания» части картинки, иначе она будет «вылезать» за пределы выделенного ей бокса. По умолчанию имеет значение *false*. Отсутствие значение у ключа clip при его упоминании эквивалентно значению *true*. Подобное поведение верно и для других логических переключателей.

» angle – поворачивает картинку на заданный угол в градусах.

» origin – определяет координаты центра, вокруг которого вращается рисунок. Кроме непосредственно координат, origin принимает и буквенные сокращения: l, b, r и t – соответствует центру вращения слева, снизу, справа и сверху. В каждом случае выбирается середина указанной стороны. Возможны комбинации, задающие углы картинки: lt, rt, rb и lb. с обозначает центр картинки.

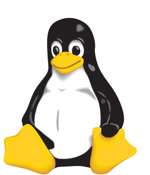
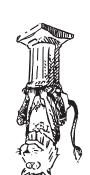
» width – ширина вставляемой картинки.

» height – высота вставляемой картинки.

» scale – масштабный коэффициент.

» keepaspectratio – логический переключатель. Модифицирует параметры высоты и ширины картинки в сторону уменьшения с целью сохранения исходных пропорций изображения.

```
\includegraphics[trim=110 0 105 100,clip,
            width=0.49\textwidth]{title.1.eps}
\hspace{0.5cm}
\includegraphics[viewport=0 0 100 200,clip,
              width=0.49\textwidth,
              height=3cm,keepaspectratio,
              angle=180,origin=c]{title.1.eps}
```

› **Ещё один пример использования \includegraphics**

Аргументы \includegraphics интерпретируются слева направо. Для команд вращения и масштабирования порядок следования имеет значение.

По вашим правилам

Пакет *graphicx* предоставляет возможность вызвать внешнюю программу для обработки картинки перед её вставкой. Так, например, можно добавить возможность включения в документ PNG-файлов:

```
\DeclareGraphicsRule{.png}{eps}{.bb}{`bmeps -p3 -c #1`}
```

Первый параметр определяет расширение нового графического формата, для которого задаются правила. В представленном примере это **.png**. Второй параметр указывает тип графики. После преобразования это будет EPS – по умолчанию, *dvips* ничего другого и не знает. Третий параметр определяет расширение файла, из которого будут считаны параметры BoundingBox. Файл должен содержать одну строчку вида:

```
%%BoundingBox: 0 0 848 979
```

Такой файл необходимо предварительно создать для каждой PNG-картинки. Это можно сделать, например, так:

```
bmeps -b «картинка».png «картинка».bb
```

Последний параметр определяет команду, которую следует выполнить для преобразования картинки. Команда должна выдавать результат в стандартный поток вывода. #1 соответствует имени обрабатываемого файла. Непосредственное выполнение команды происходит при трансляции dvi-файла.

Выполнение внешней команды является потенциально опасной процедурой, поэтому защита по умолчанию этого не позволяет. Для просмотра dvi-файла через »

[2] Отключается возможность выполнения таких команд, как удаление и переименование, а чтение файлов происходит в режиме read-only. Очень полезный ключ, если *Ghostscript* используется в качестве фильтра.

[3] The Comprehensive TeX Archive Network. Центральный сайт: http://www.ctan.org.

[4] Это стало возможным, начиная с версии PostScript Level 2.

[5] Это важно, так как мало какой растровый редактор может оптимально сохранить EPS.

[6] *graphicx* пришёл на смену пакету *graphics* — различия в последней букве. Команды из предыдущего пакета также можно использовать, но делать это настоятельно не рекомендуется.

xdvi следует использовать ключ -allowshell. В противном случае будет выдаваться запрос на выполнение команды каждый раз, когда встречается вставка по новым правилам. Для преобразования в PostScript через *dvips* также следует отключить защиту с помощью ключика -R0. Лучше всё-таки, по возможности, избегать вышеописанной процедуры и сразу готовить картинки в формате EPS.

Для трактования всех неизвестных драйверу расширений как EPS следует применить команду:

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

Это полезно в случае вставки картинок *MetaPost*, которые по умолчанию не имеют расширений. Если третий параметр равен «звёздочке», то это означает, что BoundingBox следует искать в том же файле, что и графику.

Плавающие объекты

Мало просто добавить картинку – её надо красиво разместить, и, по возможности, она должна это делать самостоятельно. Просто \includegraphics для этого дела не очень подходит, так как размещение регулируется исключительно пользователем. Для этой цели в *LaTeX* имеется специальная сущность: плавающий объект (float[7]). Если для данного объекта не хватает места на текущей странице, он переносится на следующую.

Стандартные классы определяют плавающий объект для размещения картинок как окружение figure:

```
\begin{figure}[ht]
 \centering%центрируем картинку
 \includegraphics{«картинка»}
 \caption{«подпись»}\label{fig:metka}
\end{figure}
```

В качестве необязательного параметра окружению figure можно передать допустимые способы размещения плавающего объекта:

- h – разместить по возможности здесь же,
- t – разместить в верхней части страницы,
- b – разместить в нижней части страницы,
- p – разместить на отдельной странице, где нет ничего, кроме плавающих объектов.

Приоритет размещения определяется порядком следования букв. Если первой следует буква h, то в случае неудачи *LaTeX* размещает плавающий объект на следующей странице. Если же первыми следуют буквы t или b, то размещение организуется на текущей странице.

Для «красивого» размещения картинок *LaTeX* опирается на некоторые значения по умолчанию, которые не всегда могут быть оптимальными для текущего случая. Поэтому, если очень хочется разместить картинку, например, внизу, то пожелание можно усилить с помощью восклицательного знака: [b!].

Управление плавающими объектами

Если плавающих объектов в документе немного, то всё будет хорошо и без какого-либо вмешательства человека. Но если их много, то так или иначе надо будет управлять их размещением.

clearpage

Если *LaTeX* не справляется с размещением картинок, то он переносит их на следующую страницу. В какой-то момент может накопиться целая «толпа» таких перенесённых картинок, и возникнет необходимость в их «насильственном» выводе в каком-то определённом месте. Для этого существует команда \clearpage. При вызове этой команды завершается текущая страница и выводятся все отложенные плавающие объекты – и только потом продолжается обычный вывод текста. Единственная проблема этой команды в том, что текущая страница обрывается. Чтобы избежать обрыва, можно воспользоваться пакетом *afterpage*, точнее, одноимённой командой из него:

```
\afterpage{\clearpage}
```

Команда \afterpage откладывает выполнение указанных в ней инструкций до конца текущей страницы.

suppressfloats

Команда \suppressfloats полностью подавляет размещение плавающих объектов на текущей странице. В качестве необязательного параметра ей можно передать t или b – в этом случае запрет распространяется только на размещение плавающих объектов вверху или внизу страницы, соответственно.

placeins

Пакет *placeins* не даёт плавающим объектам «утекать» за установленные пределы. Барьер устанавливается с помощью команды \FloatBarrier.

Это бывает полезно, когда хочется, чтобы все картинки не выходили за пределы своего раздела. В этом случае следует переопределить нужную команду секционирования для установки перед ней барьеров. В случае команды секционирования раздела (section) достаточно передать пакету при загрузке опцию [section]:

```
\usepackage[section]{placeins}
```

```
endfloat
```

Часто при подготовке статей необходимо размещать иллюстрации на отдельных страницах после основного текста. Подобная галерея предваряется списком иллюстраций. Пакет *endfloat* именно это и делает. Достаточно просто загрузить его.

«Упаковка» картинок в один float

Для уменьшения «поголовья» плавающих объектов полезно размещать картинки группами. Например, чтобы разместить две картинки рядом, можно применить команду \parbox или окружение minipage:

```
\parbox[«позиционирование»]{ширина}{текст}

\begin{minipage}[«позиционирование»]{ширина}
 текст
\end{minipage}
```

В обоих случаях поддерживается обязательный параметр «ширина», по которой формируется создаваемый бокс, и необязательный «позиционирование» – расположение сформированного бокса относительно базовой линии по вертикали. Позиционирование может проводиться по центру (опция [c] – задана по умолчанию), по верхней линии ([t]) и по нижней линии бокса ([b]). Шаблон для двух стоящих рядом рисунков может иметь примерно следующий вид:

```
\begin{figure}[ht]\centering
 \parbox[b]{0.49\TeXtwidth}{\centering
  \includegraphics{«рисунок-1»}
  \caption{«подпись-1»}\label{fig:metka-1}}
 \hfil\hfil%раздвигаем боксы по горизонтали
 \begin{minipage}[b]{0.49\TeXtwidth}
  \centering
  \includegraphics{«рисунок-2»}
  \caption{«подпись-2»}\label{fig:metka-2}
 \end{minipage}
\end{figure}
```

Использование команды \parbox или окружения minipage зависит исключительно от личных предпочтений. С их помощью можно организовать и более сложные конструкции.

Для целей автоматизации упаковки можно использовать специализированные пакеты:

- *subfig* – организует группы из множества картинок. Относительно современный пакет.
- *miniplot* – делает то же что *subfig*, хоть и менее изощрённо.
- *figsize* – специализируется на автоматическом вычислении размеров картинок для размещения их в указанных пределах.
- *dpfloat* – определяет новый тип плавающего окружения, занимающего сразу две страницы – двойные иллюстрации на развороте.

Картинки «в оборку»

Маленькие иллюстративные рисунки удобно делать в оборку с текстом, то есть текст должен обтекать их. Такие картинки располагаются на внешней стороне страницы – слева для чётных и справа для нечётных страниц или, в случае, одностороннего режима печати, справа.

Традиционно описываются два пакета для создания подобных рисунков: *floatflt* и *wrapfig*. *floatflt* более автоматизирован для размещения картинок, но он также чаще «ломается» при большом числе плавающих объектов. Возможны даже «потери» картинок. Упомянутые пакеты определяют окружения floatingfigure и wrapfigure, соответственно.

```
\begin{floatingfigure}[«размещение»]{«ширина»}
```

[7] Пакет, который позволяет создавать новые типы плавающих объектов, так и называется: *float*. Вместо него можно использовать *floatraw*, созданный Ольгой Лапко.

› **Картинка в оборку.**

текст текст




Рис. 1. Подпись

```
 …
\end{floatingfigure}
```

Необязательный параметр «размещение» позволяет изменить алгоритм размещения картинки:

» rflt – размещать справа,
» lflt – размещать слева,
» vflt – слева для чётных и справа для нечётных страниц (по умолчанию).

```

\begin{wrapfigure}[«число строк в оборке»]
            {«размещение»}{«ширина»}
 …
\end{wrapfigure}
```

В отличие от окружения floatingfigure, wrapfigure требует определить правила размещения картинки. Доступные варианты: справа ({r}), слева ({l}), с внешней стороны страницы ({i}) и с внутренней стороны страницы ({o}). Если вместо строчных букв передать заглавные, то включается запрет на сдвиг по вертикали – картинка должна быть размещена начиная с той строки абзаца, в которой она была определена.

Необязательный параметр «число строк в оборке» позволяет указать число строк текста, которые должны быть сбоку от картинки, при этом выносная формула считается за три строки текста. Если параметр не определён, то число строк вычисляется автоматически – к сожалению, не всегда оптимально.

Свою процедуру размещения картинки в оборку с текстом предлагает так же и пакет *nccfloats*[8]:

```
\sidefig(«ширина картинки»)(«ширина текста»)
{\includegraphics{«картинка»}}{«текст»}
```

В этом случае предлагается передавать команде \sidefig и саму картинку, и текст, помещаемый сбоку. Параметр «ширина текста» можно опустить – тогда текст занимает всё оставшееся пространство. Подробности можно посмотреть в документации **nccfloats.pdf**.

Подписи к рисункам

Для добавления подписи к рисунку используется команда \caption, которую можно использовать только внутри плавающих объектов. В качестве обязательного параметра передаётся текст подписи. При выводе подпись центрируется, если она достаточно мала. В противном случае подпись оформляется в виде абзаца. Текст подписи не должен содержать команд разрыва строки. Все хрупкие команды внутри подписи должны быть защищены с помощью команды \protect. \caption можно передать также необязательный параметр, который должен представлять из себя краткую версию подписи, появляющуюся в автоматически создаваемых списках.

Оформление подписи жёстко привязано к стилю документа, и изменить её без переопределения самой команды \caption непросто. Для модификации параметров следует воспользоваться пакетами *caption* или *ccaption*. Оба упомянутых пакета позволяют «покрутить» множество ручек и снабжены исчерпывающей документацией.

При включении русской опции \usepackage[russian]{babel} перед подписью выводится слово «**Рис.**», за которым следует автоматически вычисляемый порядковый номер картинки. В качестве разделителя между счётчиком и подписью по умолчанию используется двоеточие. Для замены двоеточия на точку в преамбуле достаточно набрать, например, следующее:

```
\usepackage{ccaption}
% заменяем для рисунков ':' после номера рисунка на '.'
\captiondelim{. } % после точки стоит пробел!
```

Кроме традиционного размещения подписи под картинками, её можно вынести, например, на поля страницы. Недаром же стандартные классы имеют такие широкие поля! Пакет *mcaption* определяет окружение margincap:

```
\begin{figure}[ht]
 \begin{margincap}{«Подпись»}
  \includegraphics{«картинка»}
 \end{margincap}
\end{figure}
```

В качестве обязательного параметра окружению передаётся подпись, а внутри определяется картинка.

Странный получился рассказ. В статье, озаглавленной «Графика», не было сказано ни слова о том, как эту графику, собственно говоря, создавать. А ведь есть что сказать, но в данном случае информация о размещении и оформлении готовых картинок поважнее будет. О том, что *LaTeX* умеет рисовать, мы тоже поговорим, но чуть попозже. LXF

OpenSource шагает по стране – «пользователи» наступают.

Посмотрите на картинку. Что это? Это *TpX* – простой векторный редактор для Windows.

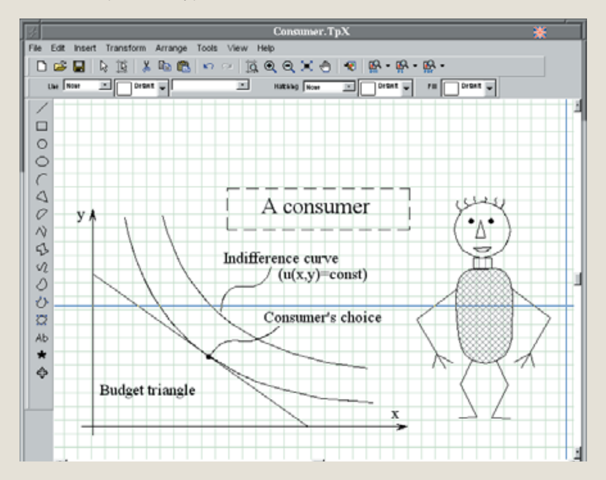
Чем же он интересен? Гм, сложный вопрос. Ну да – его можно запустить в Wine и от некоторых действий он даже не выпадает в осадок. Но это не *Inkscape* и отнюдь не *xfig*. Ещё один ничем не выдающийся «велосипед», если бы не три «но».

Первое «но» состоит в том, что этот «велосипед» выпущен под GPL: (http://sourceforge.net/projects/tpx/).

Второе «но» заключается в специализации этого редактора – он предназначен для создания простых картинок с последующим внедрением в *LaTeX*. То есть это всё-таки специализированный «велосипед».

«Но» третье и основное заключается в авторе. Это Александр Анатольевич Цыплаков (http://www.nsu.ru/ef/tsy/) – кандидат экономических наук и доцент Новосибирского государственного университета. Да, для разработки использовался Delphi, да информацию, что программа под GPL найти в коде непросто (её там нет) – отсутствие технической культуры, да, диссертация написана в *Microsoft Word* – как это принято у экономистов. Но автор не технарь – он тот самый «пользователь», о котором так любят рассуждать программисты. Пользователь, недовольный нехваткой инструментов настолько, что начал создавать их самостоятельно. Не просто создавать, а распространять результаты своего труда под свободной лицензией. И этой программой вполне можно пользоваться.

Правда, пока автор не освоит в совершенстве Lazarus, более-менее стабильной версии под Linux (у автора это в планах) вряд ли можно ожидать. Ну, естественно, если кто-нибудь ему не поможет, а то «пользователи» будут вынуждены взять власть в свои «очумелые» руки.

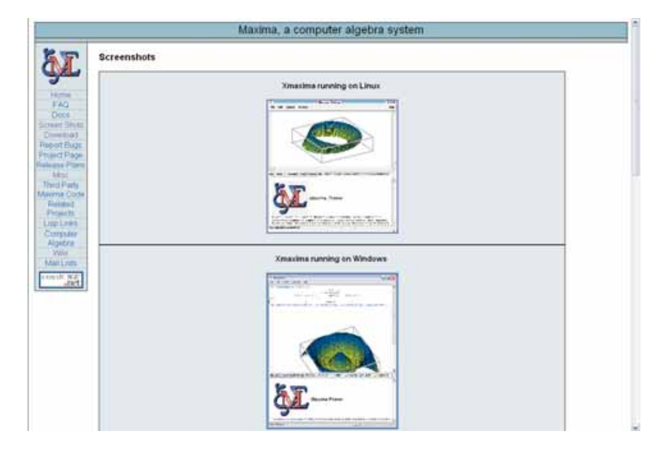


[8] *nccfloats* входит в коллекцию *ncctools*, созданную А.И. Роженко

» **Через месяц** LaTeX для программистов – представление кода и алгоритмов в печатном виде.

Работа с файлами

ЧАСТЬ 6 Завершая этот длинный цикл статей, **Тихон Тарнавский** коснется вопросов работы с файлами, базой данных фактов и напишет собственную функцию символьного дифференцирования!



В прошлый раз мы остановились на возможностях программирования, предназначенных для написания собственных функций и модулей к *Maxima* – и теперь для их полноценного использования рассмотрим несколько инструментов работы с файлами, позволяющих сохранять и загружать эти функции и модули на диск и с диска. Далее речь пойдет о наложении определенных условий на неизвестные и значения функций. Напоследок познакомимся с функциями по работе... с функциями: это один из очень мощных инструментов, позаимствованных из функционального программирования; а также разберем несколько более крупных учебных примеров, использующих многое из изученного нами во всех статьях цикла.

Учимся читать и писать

Среди средств для операций с файлами функции с наиболее очевидными именами – save и load – имеют, вопреки привычной для *Maxima* логичности всех названий, различный контекст. Первая предназначена для выгрузки *Maxima*-выражений в виде исходных кодов на Lisp, так что если вы не знаток Lisp (да и реализации внутренних механизмов *Maxima*), то эта функция представляет лишь чисто академический интерес. Посему подробнее мы займемся другими функциями – для обработки так называемых пакетных (batch) файлов, хранящих выражения уже в синтаксисе самой *Maxima*. А поскольку в виде таких файлов поставляется немалое количество функционала *Maxima*, то начнем с загрузки. И вот о второй из очевидно-именуемых функций здесь уже будет рассказано.

Функции чтения файлов с выражениями *Maxima* существует три: demo(имя-файла), batch(имя-файла) и batchload(имя-файла). Первая предназначена для загрузки так называемых демо-файлов, задуманных, как и явствует из названия, для демонстрационных примеров. Она загружает демо-файл и выполняет его в пошаговом режиме, ожидая нажатия Enter после выполнения каждой строки. В составе *Maxima* поставляется значительное количество демо-файлов; упоминания о них можно найти в документации, а сами файлы несложно обнаружить среди содержимого пакета *maxima-share* (либо, в случае отсутствия такового в вашем дистрибутиве, просто *maxima*) по их расширению – **.dem**.

Функция batch() загружает *Maxima*-файл с расширением **.mac** или **.mc** (от первоначального названия программы – *Macsyma*) и выполняет содержащиеся в нем выражения так, как если бы они вводились прямо в текущей сессии, то есть с отображением результата каждого выражения и назначением меток %iN, %oN. Функция batchload(), напротив, подгружает пакетный файл «молча»: все назначенные в нем функции и переменные становятся доступны, но результаты не видны, и весь хранимый ввод-вывод, включая значения символов % и _ и результаты, возвращаемые функцией %th(), остается тем же, что и до вызова.

Функции batch() и batchload() используют при поиске файлов для загрузки путь (точнее сказать, шаблон, потому как в нем содержатся не только имена каталогов, но и допустимые расширения файлов), который хранится в переменной file_search_maxima. По умолчанию эта переменная содержит все каталоги, в которые устанавливаются **.mac**-файлы из пакетов *Maxima*, а также **~/.maxima**, предназначенный для пользовательских файлов. Для других функций загрузки существуют отдельные переменные: file_search_lisp и file_search_demo, смысл которых понятен из их названий.

Ну и под конец здесь нужно вспомнить о вышеназванной функции load. Она, фактически, является оберткой над двумя функциями: уже описанной выше batchload() и loadfile(), вторая, совершенно аналогично первой, загружает файл, но уже не с выражениями *Maxima*,

Месяц назад Мы научились строить графики и писать программы.

и фактами

а с исходным кодом Lisp, то есть является парной к функции save(). Функцию load() можно, в принципе, использовать вместо batchload(): путь file_search_maxima задан в ней раньше, чем file_search_lisp, так что в случае неоднозначности она будет загружать файлы *Maxima*; а кроме того, так короче.

Некоторый функционал *Maxima* содержится в неподгружаемых автоматически внешних файлах, которые, соответственно, нужно принудительно загрузить перед использованием:

(%i1) A: matrix$([a_{11}, a_{12}, a_{13}], [a_{21}, a_{22}, a_{23}])$

(%o1) $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$

(%i2) matrix_size(A)

(%o2) matrix`size$\left(\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}\right)$

(%i3) load(linearalgebra)

(%o3) /usr/share/maxima/5.10.0/share/linearalgebra/linearalgebra.mac

(%i4) ''%i2

(%o4) [2, 3]

Помимо ручной загрузки нужного файла, можно также настроить *Maxima* на автоматическую подгрузку в случае вызова заданной функции. Делается это так: setup_autoload(имя-файла, имена-функций); нужные функции здесь перечисляются через запятую прямо после имени файла. Удобнее, конечно, будет не вызывать функцию setup_autoload() вручную (так в ней и толку немного), а настроить *Maxima* на автоматический ее запуск при старте программы. Файл, который, при его наличии, вызывается при каждом запуске *Maxima*, называется **maxima-init.mac** и самое логичное для него местоположение – все тот же каталог **~/.maxima**. Конечно, он может содержать не только вызовы функции setup_autoload(), а любые выражения *Maxima*, которые вы хотите выполнять при каждом ее запуске. Использование этой функции может сделать вашу работу с *Maxima* намного более удобной в том случае, если вы часто используете некоторые из внешних функций *Maxima* или функции, вами же и написанные.

Для полноценного чтения файлов всего сказанного уже вполне достаточно, теперь перейдем к записи в них. Тут нас в первую очередь интересует функция stringout(), которая позволяет выгружать в файл любые выражения и функции *Maxima* в точно таком виде, в каком их загружают функции demo(), batch() и batchload(). С ее помощью можно писать выражения, которые вы хотите иметь во внешнем модуле, находясь непосредственно в интерфейсе *Maxima*, с последующей записью в этот самый модуль. Для выгрузки функций в один из стандартных каталогов *Maxima* (самым логичным вариантом будет, пожалуй, упомянутый выше **~/.maxima**) имя файла во всех вариантах вызова функции stringout() нужно задавать с полным путем; в случае задания имени без пути файл будет создан в текущем каталоге, то есть в том, откуда производился запуск *Maxima*.

Здесь, чтобы было интереснее и не приходилось писать в файлы всякую ерунду, немного прервемся и создадим пару небольших функций.

(%i1) primes(n) :=
 if integerp(n) then
 if $n \leqslant 2$ then [] else
 append(primes(prev_prime(n)), [prev_prime(n)])$

Эта функция возвращает список всех простых чисел, меньших чем заданное целое число. Сначала мы проверяем, является ли аргумент целым числом и делаем это простейшим образом: в случае невыполнения условия оператор if, напомню, вернет *false*. Генерируется список тоже самым простым и коротким в реализации способом – рекурсией. (примечание для людей, далеких от программирования: рекурсивная функция – это функция, вызывающая саму себя; чаще всего такие функции строятся по принципу индукции). Здесь используется функция *Maxima* по имени prev_prime(), которая возвращает простое число, предшествующее заданному целому.

У рекурсии, при всей ее простоте реализации, есть неоспоримый минус – только один, но весьма существенный: чрезвычайная требовательность к объему памяти. Поэтому, для обеспечения возможности получать последовательности из больших простых чисел, добавим в наш учебный пример еще одну функцию:

(%i2) primesbetween(n, m) :=
 if integerp(n) and integerp(m) then
 if $m \leqslant 2$ or prev_prime(m) $\leqslant n$ then [] else
 append(primesbetween(n, prev_prime(m)),
 [prev_prime(m)])$

Смысл, думаю, понятен по аналогии с предыдущей: теперь мы еще и ограничили возвращаемый список снизу.

Теперь, когда у нас уже есть primesbetween(), первую функцию можно написать по «принципу чайника» – сведя задачу к предыдущей:

(%i3) primes1(n) := primesbetween(1, n)$

Теперь вернемся к stringout(). Эта функция, как и многие другие, может принимать несколько различных вариантов аргументов, первым из которых всегда выступает имя файла для записи, а остальные отвечают за то, что же именно будет туда записано. В варианте stringout(имя-файла, [начало, конец]) записаны будут ячейки ввода с номерами от «начала» до «конца» включительно:

»

»

(%i4) stringout(".maxima/primes.mac", [1, 2])

(%o4) /home/t/.maxima/primes.mac

```
$ cat .maxima/primes.mac
primes(n):=if integerp(n) then (if n <= 2 then [] else append(primes(prev_prime(n)),[prev_prime(n)]));
primesbetween(n,m):=if integerp(n) and integerp(m) then (if m <= 2 or prev_prime(m) <= n then [] else append(primesbetween(n,prev_prime(m)),[prev_prime(m)]));
```

Как видите, по умолчанию вывод получается не слишком красивым, поэтому сразу рассмотрим один ключ, влияющий на его формат. Долго рассказывать о нем смысла нет, лучше показать на примере:

(%i5) grind: true$ stringout(".maxima/primes.mac", [1, 2])

(%o6) /home/t/.maxima/primes.mac

```
$ cat .maxima/primes.mac

primes(x):=if integerp(x)
      then (if x <= 2 then []
              else append(primes(prev_prime(x)),[prev_prime(x)]));
primesbetween(n,m):=if integerp(n) and integerp(m)
           then (if m <= 2 or prev_prime(m) <= n then []
                   else append(primesbetween(n,prev_prime(m)),
                         [prev_prime(m)]));
```

Представления о правилах отступов у создателей этой опции несколько специфичные, но тем не менее, результат стал намного читабельнее. Так что, если вы планируете сохранять выражения *Maxima* не только для того, чтобы потом загружать их обратно, а желаете редактировать созданные файлы, я рекомендую вам прописать grind:true глобально в файле **~/.maxima/maxima-init.mac**.

Идем дальше. С помощью ключевого слова input можно выгрузить в файл все ячейки ввода разом:

(%i7) $(N: [\mathrm{random}(100000)],$
for i thru 9 do
$N: \mathrm{append}(N, [N[i] + \mathrm{random}(100000)]),$
$N)$

(%o7) [49900, 61971, 153219, 244360, 290427, 347723, 396481, 465378, 522906, 568462]

(%i8) $(P: [],$
for i thru 10 do
$P: \mathrm{append}(P, \mathrm{primesbetween}(N[i] - 50, N[i])),$
$P)$

(%o8) [49853, 49871, 49877, 49891, 61927, 61933, 61949, 61961, 61967, 153191, 244313, 244333, 244339, 244351, 244357, 290383, 290393, 290399, 290419, 347707, 347717, 396437, 396443, 396449, 396479, 465331, 465337, 465373, 522857, 522871, 522881, 522883, 522887, 568433, 568439, 568441, 568453]

(%i9) stringout("primes-sample.mac", input)

(%o9) /home/t/primes-sample.mac

```
$ cat primes-sample.mac

primes(n):=if integerp(n)
      then (if n <= 2 then []
              else append(primes(prev_prime(n)),[prev_prime(n)]));
primesbetween(n,m):=if integerp(n) and integerp(m)
           then (if m <= 2 or prev_prime(m) <= n then []
                   else append(primesbetween(n,prev_prime(m)),
                         [prev_prime(m)]));
primes1(n):=primesbetween(1,n);
stringout(".maxima/primes.mac",[1,2]);
grind:true;
stringout(".maxima/primes.mac",[1,2]);
(N:[random(100000)],for i thru 9 do N:append(N,[N[i]+random(100000)]),N);
(P:[],for i thru 10 do P:append(P,primesbetween(N[i]-50,N[i])),P);
```

Кроме input, есть еще два ключевых слова: functions и values. Первое позволяет записать определения всех функций, второе – присвоение всем символам выражений их текущих значений:

(%i10) stringout(".maxima/primes.mac", functions)

(%o10) /home/t/.maxima/primes.mac

(%i11) stringout("primes-sample.mac", functions, values)

(%o11) /home/t/primes-sample.mac

```
$ cat .maxima/primes.mac

primes(n):=if integerp(n)
      then (if n <= 2 then []
              else append(primes(prev_prime(n)),[prev_prime(n)]));
primesbetween(n,m):=if integerp(n) and integerp(m)
           then (if m <= 2 or prev_prime(m) <= n then []
                   else append(primesbetween(n,prev_prime(m)),
                         [prev_prime(m)]));
primes1(n):=primesbetween(1,n);


$ cat primes-sample.mac

primes(n):=if integerp(n)
      then (if n <= 2 then []
              else append(primes(prev_prime(n)),[prev_prime(n)]));
primesbetween(n,m):=if integerp(n) and integerp(m)
           then (if m <= 2 or prev_prime(m) <= n then []
                   else append(primesbetween(n,prev_prime(m)),
                         [prev_prime(m)]));
primes1(n):=primesbetween(1,n);
N:[49900,61971,153219,244360,290427,347723,396481,465378,522906,568462];
P:[49853,49871,49877,49891,61927,61933,61949,61961,61967,153191,244313,244333,
  244339,244351,244357,290383,290393,290399,290419,347707,347717,396437,
  396443,396449,396479,465331,465337,465373,522857,522871,522881,522883,
  522887,568433,568439,568441,568453];
```

И кроме всего этого, функцию stringout() можно вызвать с непосредственным перечислением в аргументах конкретных выражений. В этом случае, надо заметить, будут сохраняться не ячейки, содержащие заданные выражения, а именно сами выражения. То есть, если перечислить символ, для которого задано значение, то в файл будет записано только это значение. С именами функций, заданными непосредственно, дело обстоит не лучше: функцию таким образом задать, по сути, вообще нельзя: если просто написать ее имя, то вместо функции будет подставлен одноименный символ (или его значение, если оно задано). Но из обеих ситуаций есть выход. Для функций – штатный: функция fundef, которая принимает имя любой пользовательской функции и возвращает ее определение в точности в таком же виде, в каком оно было введено (или могло бы быть введено) в «командной строке» *Maxima*, с точностью до пробелов:

(%i12) stringout(".maxima/primesbetween.mac", fundef(primesbetween))

(%o12) /home/t/.maxima/primesbetween.mac

(%i13) stringout(".maxima/primes1.mac", fundef(primes), fundef(primes1))

(%o13) /home/t/.maxima/primes1.mac

```
$ cat .maxima/primesbetween.mac

primesbetween(n,m):=if integerp(n) and integerp(m)
          then (if m <= 2 or prev_prime(m) <= n then []
                 else append(primesbetween(n,prev_prime(m)),
                        [prev_prime(m)]));


$ cat .maxima/primes1.mac

primes(n):=if integerp(n)
     then (if n <= 2 then []
          else append(primes(prev_prime(n)),[prev_prime(n)]));
primes1(n):=primesbetween(1,n);
```

А для символов можно использовать небольшую хитрость: блокировать вычисление переданного выражения, а в нем написать сначала сам символ, а потом через двоеточие – его же, предварив знаком принудительного вычисления (два апострофа):

(%i14) stringout("random-primes.mac",'(P:''P))

(%o14) /home/t/random-primes.mac

```
t:~$ cat random-primes.mac

P:[49853,49871,49877,49891,61927,61933,61949,61961,61967,153191,
244313,244333,
  244339,244351,244357,290383,290393,290399,290419,347707,34771
7,396437,
  396443,396449,396479,465331,465337,465373,522857,522871,52288
1,522883,
  522887,568433,568439,568441,568453];
```

В довершение темы работы с файлами стоит обратить внимание еще на один момент: при загрузке файлы в текущем каталоге не ищутся – и как раз для него надо задавать путь, причем полный, а не через **./имя-файла**:

```
(%i1) load("primes-sample")

Could not find 'primes-sample' using paths in
file_search_maxima,file_search_lisp (combined values:
[/home/t/.maxima/###.mac,mc,
/usr/share/maxima/5.10.0/share/###.mac,mc,
/usr/share/maxima/5.10.0/share/affine,algebra,algebra/charsets
/home/t/.maxima/###.o,lisp,lsp,
/usr/share/maxima/5.10.0/share/###.o,lisp,lsp,
/usr/share/maxima/5.10.0/share/affine,algebra,algebra/charsets
/usr/share/maxima/5.10.0/src/###.o,lisp,lsp] )
 -- an error.  Quitting.  To debug this try
debugmode(true);

(%i2) load(primes)

(%o2) /home/t/.maxima/primes.mac

(%i3) load("/home/t/primes-sample")

(%o3) /home/t/primes-sample.mac

(%i4) load("./primes-sample")

Could not find './primes-sample' using paths in
file_search_maxima,file_search_lisp (combined values:
```

«Прослушайте объявление»

Теперь поговорим о функциях, позволяющих налагать определенные условия на выражения, которыми оперирует *Maxima*. Таких функций существует две, и достаточно разноплановых; но определенная связь между ними есть, так как все условия, заданные ими на данный момент, хранятся в общей «базе». Первая из этих функций называется declare (объявлять). С ее помощью можно объявлять весьма разнообразные факты о произвольных символах или выражениях; синтаксис ее весьма прост: declare(имя, факт) или declare(имя1, факт1, имя2, факт2, ...); факты задаются с помощью ключевых слов. Сами факты я бы разделил на три группы: «технические» факты *Maxima*, позволяющие использовать наделенный ими символ некоторым специальным образом при вводе выражений; факты о символах (атомарных выражениях); и факты о значениях функций. К первым относятся, к примеру, свойства evflag и evfun, о которых шла речь в описании функции ev; некоторые штатные функции обладают ими по умолчанию, а с помощью функции declare мы можем присвоить эти свойства любым другим, в том числе и пользовательским, функциям. Вторая группа фактов несет информацию о неизвестных; например, мы можем указать, что некоторая неизвестная является константой, или что ее значение – целое. И третья группа – примерно то же самое, но о функциях; примеры: четная функция (f(–x)=f(x)), аддитивная (f(x+y)=f(x)+f(y)) или целочисленная. Для краткости просто перечислим наиболее интересные из возможных фактов, сгруппировав соответственно трем упомянутым группам.

Технические факты

evfun

Позволяет применять функцию или переменную как опцию, то есть «выражение, имя-функции» вместо «имя-функции(выражение)» или «выражение, имя-переменной» вместо «имя-переменной:true; выражение». Подробнее см. в LXF82.

bindtest

Запрещает использовать символ в выражениях до присвоения ему значения. При таком использовании *Maxima* выдаст ошибку. Пример см. в документации.

feature

Делает заданное имя именем свойства (факта), что дает возможность использовать его точно так же, как все перечисленные здесь имена.

Факты о символах

constant

Имя трактуется как константа.

scalar

Имя трактуется как скалярная величина. На это также влияет флаг assumescalar: если он равен *true*, то все неопределенные символы воспринимаются как скаляры. Тут есть небольшая коллизия: если верить документации, то по умолчанию assumescalar равен *false*, реально же в *Maxima 5.10.0* он равен *true*.

(%i1) $v: [v1, v2, v3]\$$ assumescalar: false$

(%i3) $N v$

(%o3) $N\,[v1, v2, v3]$

(%i4) declare(N, scalar); $N v$

(%o4) done

(%o5) $[N\,v1, N\,v2, N\,v3]$

nonscalar

Имя трактуется как не-скалярная величина, то есть матрица или вектор.

integer, noninteger

Целое и нецелое число.

even, odd

Четное и нечетное целое число.

(%i1) $\mathrm{declare}(n, \mathrm{even})\$$
(%i2) $\mathrm{askinteger}(n)$
(%o2) yes

Факты о функциях

rassociative

Объявляет функцию как «ассоциативную» по правому аргументу.

lassociative

Аналогично – по левому аргументу.

(%i1) $\mathrm{declare}(f, \mathrm{rassociative})\$$
(%i2) $f(a,b,c,d); f(f(a,b), f(c,d))$
(%o2) $\mathrm{f}(\mathrm{a}, \mathrm{f}(\mathrm{b}, \mathrm{f}(\mathrm{c}, \mathrm{d})))$
(%o3) $\mathrm{f}(\mathrm{a}, \mathrm{f}(\mathrm{b}, \mathrm{f}(\mathrm{c}, \mathrm{d})))$
(%i4) $\mathrm{declare}(f, \mathrm{lassociative})\$$
(%i5) $f(a,b,c,d); f(f(a,b), f(c,d))$
(%o5) $\mathrm{f}(\mathrm{f}(\mathrm{f}(\mathrm{a}, \mathrm{b}), \mathrm{c}), \mathrm{d})$
(%o6) $\mathrm{f}(\mathrm{f}(\mathrm{f}(\mathrm{a}, \mathrm{b}), \mathrm{c}), \mathrm{d})$
(%i7) $\mathrm{f}(\mathrm{a}, \mathrm{f}(\mathrm{b}, \mathrm{f}(\mathrm{c}, \mathrm{d})))$
(%o7) $\mathrm{f}(\mathrm{f}(\mathrm{f}(\mathrm{a}, \mathrm{b}), \mathrm{c}), \mathrm{d})$

nary

Объявляет «**n**-арную» функцию. Это и два предыдущих названия не совсем точны: **n**-арной правильно называть функцию от **n** аргументов, а лево- и право- ассоциативной – функции именно с односторонней ассоциативностью, то есть, для «лево-» f(f(a,b),c)≠f(a,b,c)≠f(a,f(b,c)). А в *Maxima* все три факта объявляют на самом деле полно-ассоциативную функцию от произвольного числа аргументов, а различаются только тем, как будут расставлены скобки по умолчанию.

(%i8) $\mathrm{kill}(f)\$ \ \mathrm{declare}(f, \mathrm{nary})\$$
(%i10) $''\%i7$
(%o10) $\mathrm{f}(\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d})$

symmetric/commutative

Оба ключевых слова объявляют функцию как симметричную (коммутативную).

(%i1) $\mathrm{declare}(f, \mathrm{symmetric})\$$
(%i2) $f(a,b) + f(b,a)$
(%o2) $2\,\mathrm{f}(\mathrm{a}, \mathrm{b})$

antisymmetric

Объявляет функцию как антисимметричную.

(%i1) $\mathrm{declare}(f, \mathrm{antisymmetric})\$$
(%i2) $f(a,b) + f(b,a)$
(%o2) 0

outative

Константа выносится за знак функции.

(%i1) $\mathrm{declare}(f, \mathrm{outative}, N, \mathrm{constant})\$$
(%i2) $f(N x)$
(%o2) $\mathrm{N}\,\mathrm{f}(\mathrm{x})$

Многие из фактов, которые можно устанавливать с помощью функции declare, сохраняются в «базе данных» фактов. Узнать текущее состояние этой базы можно с помощью функции facts(). Ее можно вызывать, либо передав в качестве единственного аргумента имя, список фактов по которому мы хотим получить, либо вообще без аргументов – тогда будут выданы все известные факты обо всех пользовательских именах. Удалить свойства позволяет функция remove(). Она, как и многие другие, имеет несколько вариантов вызова. Будучи вызвана как remove(имя, свойство) или remove(имя1, свойство1, имя2, свойство2, ...), она лишает каждое переданное имя одного соответствующего ему свойства. Можно также передавать ей списки имен и свойств: remove([имя1, имя2, ...], [свойство1, свойство2, ...]); тогда каждое имя из списка будет лишено всех перечисленных свойств. Пар списков тоже может быть более одной: remove(список-имен1, список-свойств1, список-имен2, список-свойств2, ...) – этот вызов аналогичен последовательным remove(список-имен1, список-свойств1); remove(список-имен2, список-свойств2); ... И последний интересующий нас вариант – remove(all, свойство) удаляет «свойство» со всех имен, у которых оно есть.

Вторая «условная» функция – это функция assume() (допускать, принимать). Здесь все проще: в качестве аргументов ей можно передавать в любом количестве самые обыкновенные равенства и неравенства. Напомню только, что задавать их нужно не в синтаксической, а в логической форме, то есть не «a=b», «a#b», а «equal(a,b)», «not equal(a,b)». Из логических операторов допускается также использование and (по сути assume(x>0 and x<1) это то же самое, что и assume(x>0, x<1)), но не or – база фактов не поддерживает информацию вида «или»; и речь не о синтаксисе, а именно о конструкциях, то есть выражения типа not(a>b and a<c) тоже недопустимы. Факты, добавленные assume(), также видны функции facts():

(%i1) $\mathrm{declare}(n, \mathrm{integer})\$ \ \mathrm{assume}(n > 10)\$$
(%i3) $\mathrm{facts}(n)$
(%o3) $[\mathrm{kind}(\mathrm{n}, \mathrm{integer}), \mathrm{n} > 10]$

Ключевое слово kind используется только для отображения тех фактов из базы, которые добавлены с помощью declare().

Если факты, заданные функцией declare(), удаляются вызовом remove(), то для assume() есть своя «обратная» функция – forget(), которая также принимает любое количество условий (точно таких же как и assume()), либо в качестве отдельных аргументов, либо списком.

Общая база фактов используется этими двумя не очень похожими функциями неспроста: все, кому эти факты могут пригодиться, используют обе их разновидности, причем одновременно. Например, уже известный нам предикат is:

```
(%i1) declare(f,increasing)$
(%i2) assume(x > y)$
(%i3) is(f(x) > f(y))
(%o3) true
```

Еще один пример использования assume()/declare() – возможность избежать неопределенностей. Вы, возможно, помните, как в одном из примеров LXF84 в ответ на попытку посчитать некий интеграл Maxima задала нам вопрос о знаке входящего в него символа. Вот в таких ситуациях тоже может пригодиться assume, дабы предвосхитить расспросы:

(%i1) $\mathrm{integrate}(x^2\sqrt{a^2-x^2},x,0,a)$

Is a positive, negative, or zero?p

(%o1) $\frac{\pi a^4}{16}$

(%i2) $\mathrm{assume}(a<0)\$$

(%i3) $''\%i1$

(%o3) $-\frac{\pi a^4}{16}$

(%i1) $\mathrm{limit}(x^\infty)$

Is $|x|-1$ positive, negative, or zero?n

Is x positive, negative, or zero?n

(%o1) 0

(%i2) $\mathrm{assume}(x>0,x<1)\$$

(%i3) $\mathrm{limit}(x^\infty)$

(%o3) 0

Вот мы и подошли к концу «теоретической» части. Надеюсь, функционала, рассмотренного на протяжении шести статей, будет достаточно для решения многих задач, а также для того, чтобы черпать дальнейшие сведения из документации – ведь мы уже изучили такие вещи, благодаря которым *Maxima* становится не просто «вычислялкой» отдельных небольших примеров, а настоящей «средой программирования с математическим уклоном», позволяющей создавать свои собственные математические «типы данных» – числовые системы, функционалы и прочая и прочая – и полноценные программные модули, которые могут использовать весь встроенный (или также собственноручно достроенный) функционал *Maxima*. Рассмотрим, напоследок, более серьезный учебный пример, в котором эти возможности можно будет лучше прочувствовать. Одна заявленная тема у нас пока осталась нераскрытой – функции для работы с функциями и «глубокой» обработки выражений. Но это настолько серьезный инструмент, что на маленьких примерах его рассматривать было бы бессмысленно, а потому мы поговорим о нем в приложении-практикуме. Удачи! LXF

Практикум Maxima



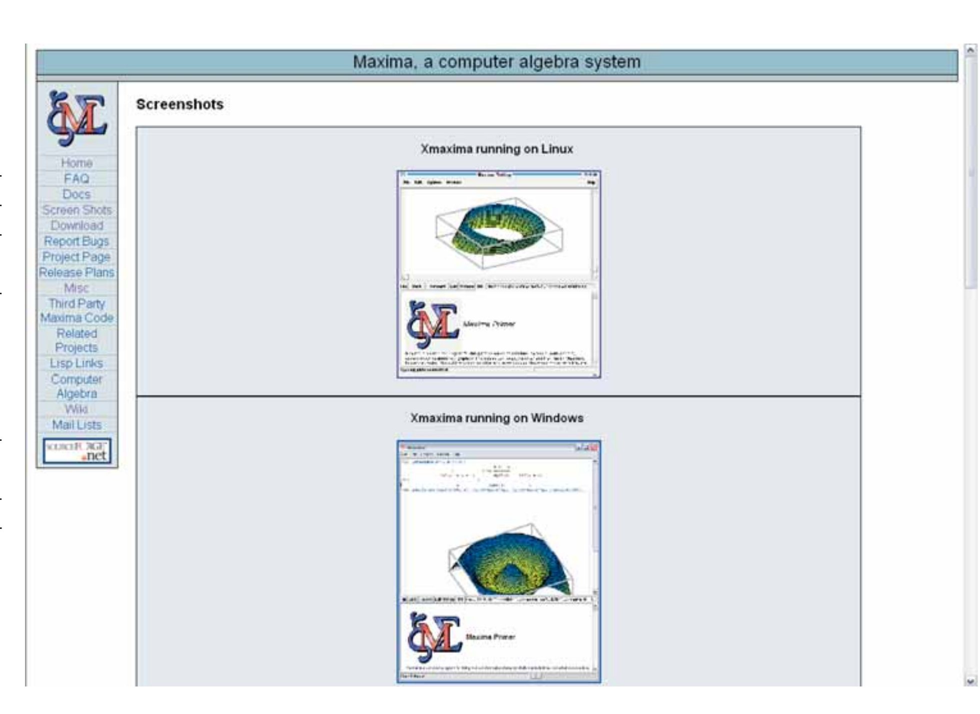
Сначала я хотел рассмотреть несколько отдельных практических примеров: и маленьких, и чуть побольше. Но потом мне подумалось, что один, но более серьезный пример будет значительно лучше: с одной стороны, его можно строить понемногу, отрабатывая отдельные приемы точно так же, как это было бы сделано и с меньшими примерами, а с другой – в результате все эти приемы переплетутся между собой во что-то объемное, и на этих переплетениях возникнет более цельное ощущение возможностей программы, чем на несвязанных маленьких кусочках. К тому же по ходу дела мы соорудим несколько небольших вспомогательных функций, а заодно, для дополнительной практики, и более расширенную версию одной из них, которая, вполне возможно, пригодится вам и в дальнейшем.

А писать мы будем настоящую функцию дифференцирования. Практически такую же, как встроенная diff(), только без вычисления полного дифференциала – чтобы не слишком сложно было «охватить» пониманием сразу весь пример. Ну а если будет интерес, то дописать вычисление полного дифференциала к этой же функции вы можете попробовать самостоятельно – после освоения возможностей, которые сейчас будут продемонстрированы, это будет уже несложно. Примеров применения по ходу создания функции я давать не буду. Если вы хотите смотреть на практические результаты, по мере добавления кода можно сохранять его в файле, скажем, **~/.maxima/deriv.mac** и выполнять в *Maxima* строку load(deriv)$ deriv(какое-нибудь-выражение);.

Я буду писать код постепенно и по ходу написания давать комментарии к последнему написанному участку. Комментировать буду, просто вставляя куски кода в текст. К слову: *Maxima* поддерживает комментарии в коде «в стиле Си», то есть комментарий начинается символами /*, а заканчивается */. Причем, в отличие от Си, допускаются вложенные комментарии: /* вот /* такие */ */.

Чтобы не повторять каждый раз весь код от самого начала, я буду сокращать его с помощью многоточия. Если вы будете проверять код по мере чтения, не забывайте о разделяющих запятых после последних строк предыдущих участков.

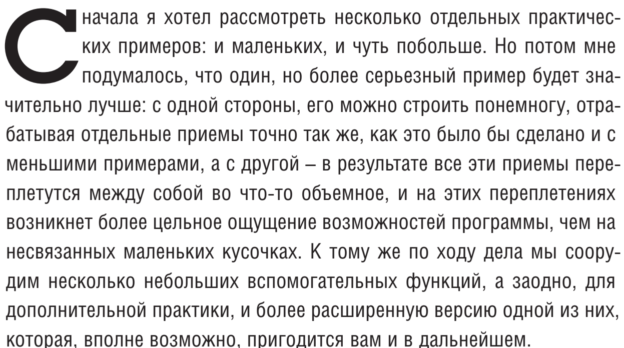
Полный текст практикума вы найдете на LXFDVD в разделе «Журнал».





Пишем свой diff()

БОНУС В этом приложении-практикуме Тихон Тарнавский покажет, как использовать Maxima для решения «настоящих» задач.



Сначала я хотел рассмотреть несколько отдельных практических примеров: и маленьких, и чуть побольше. Но потом мне подумалось, что один, но более серьезный пример будет значительно лучше: с одной стороны, его можно строить понемногу, отрабатывая отдельные приемы точно так же, как это было бы сделано и с меньшими примерами, а с другой – в результате все эти приемы переплетутся между собой во что-то объемное, и на этих переплетениях возникнет более цельное ощущение возможностей программы, чем на несвязанных маленьких кусочках. К тому же по ходу дела мы соорудим несколько небольших вспомогательных функций, а заодно, для дополнительной практики, и более расширенную версию одной из них, которая, вполне возможно, пригодится вам и в дальнейшем.



А писать мы будем настоящую функцию дифференцирования. Практически такую же, как встроенная diff(), только без вычисления полного дифференциала – чтобы не слишком сложно было «охватить» пониманием сразу весь пример. Ну а если будет интерес, то дописать вычисление полного дифференциала к этой же функции вы можете попробовать самостоятельно – после освоения возможностей, которые сейчас будут продемонстрированы, это будет уже несложно. Примеров применения по ходу создания функции я давать не буду. Если вы хотите смотреть на практические результаты, по мере добавления кода можно сохранять его в файле, скажем, **~/.maxima/deriv.mac** и выполнять в *Maxima* строку load(deriv)$ deriv(какое-нибудь-выражение);.

Я буду писать код постепенно и по ходу написания давать комментарии к последнему написанному участку. Комментировать буду, просто вставляя куски кода в текст. К слову: *Maxima* поддерживает комментарии в коде «в стиле Си», то есть комментарий начинается символами /*, а заканчивается */. Причем, в отличие от Си, допускаются вложенные комментарии: /* вот /* такие */ */.

Чтобы не повторять каждый раз весь код от самого начала, я буду сокращать его с помощью многоточия. Если вы будете проверять код по мере чтения, не забывайте о разделяющих запятых после последних строк предыдущих участков.

Начнем с «подготовительных работ»: проверки определенных условий и сохранения нужных значений в локальных переменных.

```
deriv([l]):=block([f,len,x],
        len:length(l),
        if len=0 then
        error("deriv can't be used without arguments"),
        f:l[1],
        x:listofvars(f)
)$
```

Итак, по порядку. Символ в квадратных скобках означает, что ему будет присвоен список из всех аргументов, с которыми вызвана функция. Эта конструкция предназначена для создания функций с переменным числом аргументов.

Функция block() – это расширенный аналог составного оператора. Отличается она двумя вещами. Во-первых, поддерживается возврат значений через return(), точно так же как из цикла, то есть по return(выражение) будет осуществлен выход из блока и результатом вычисления блока станет «выражение». А во-вторых, в блоке можно использовать локальные переменные – то есть такие, которые не повлияют на значения символов вне блока, даже если будут иметь совпадающие с ними имена. Такие локальные символы перечисляются в виде списка в самом начале блока.

Далее мы сохраняем в одной из таких локальных переменных длину списка аргументов (функция length) и в случае, если она равна нулю (то есть аргументов нет), генерируем ошибку функцией error, которая может принимать произвольное число аргументов, которые она вычисляет и выводит прежде чем создать ошибку.

Функция listofvars возвращает список переменных переданного ей выражения. Этот список понадобится нам для небольшого расширения возможностей: так как мы не будем вычислять полный дифференциал, то вызов с одним аргументом у нас освобождается, и мы будем его использовать аналогично функции solve: если переданное выражение включает в себя только одну неизвестную, будем дифференцировать его по ней. Продолжаем:

```
deriv([l]):=block([f,len,x],
...
    x:listofvars(f),
    if len=1 then (
        if length(x)=0 then
            return(0),
        if length(x)>1 then
            error("Expression has more than one unknowns and none was specified.","Unknowns given:", x),
        x:x[1] )
    else
        x:l[2]
)$
```

Если параметр дифференцирования в списке аргументов не задан, то проверяем длину списка неизвестных. Если она равна нулю – то это

константа и следовательно возвращаем ноль. Если больше единицы, то неизвестно, по чему дифференцировать, следовательно, снова генерируем ошибку. Ну а в случае единицы, просто превращаем список из одного элемента в сам этот элемент. Если же список аргументов длиннее, то берем параметр оттуда.

```
deriv([l]):=block([f,len,x],
...
    else
        x:l[2]
    if len>=3 then
    error("More than 2 arguments not implemented yet.")
)$
```

Пока ограничимся производной первого порядка по одной переменной. Когда этот этап будет пройден, остальное будет уже нетрудно написать на основе имеющегося кода. Теперь, когда проверки закончены, приступаем непосредственно к реализации. Строить эту функцию мы будем поэтапно. Для начала научим ее дифференцировать просто переменную и константу:

```
deriv([l]):=block([f,len,x],
...
    error("More than 2 arguments not implemented yet.")
    if atom(f) or subvarp(f) then
        if f=x then
            return(1)
        else
            return(0),
    else
        return 'diff(f,x)
)$
```

Предикат atom() проверяет, является ли его аргумент атомарным выражением, то есть константой (целой либо с плавающей точкой) или одиночным символом. Второй предикат – subvarp() – расшифровывается как subscripted variable (predicate), где первые два слова означают «индексированная переменная», то есть что-то вида **a[1]**. Добавлен этот предикат в эту же проверку в связи с тем, что *Maxima* такие выражения атомарными не считает, а с точки зрения дифференцирования они как раз являются атомами. Дальше в этом варианте все просто: если атомарное выражение является параметром дифференцирования, то результат будет равен единице, иначе – нулю: в полном соответствии с правилами дифференцирования.

В самом конце функции добавляем строку, которая в нештатном случае (таком, который мы еще не посчитали) просто вернет несовершенную форму производной от оставшегося выражения. Эта строка у нас вплоть до самой полной реализации будет оставаться последней, а все остальное мы будем вписывать до нее, сокращая тем самым этому некрасивому умолчательному случаю шансы на выживание. А двигаться дальше мы будем достаточно интересным способом, с помощью уже упомянутой в статье рекурсии. Мы будем постепенно обучать нашу функцию все новым и новым трюкам (точнее, правилам дифференцирования), разбивая неизвестные выражения некоторыми способами на более простые, уже обработанные варианты; то есть действуя снова по известному «принципу чайника». И вы увидите, что математики не зря так любят этот принцип: с его помощью такая, на первый взгляд, сложная задача будет разбита на множество простых подзадачек и таким образом упростится сама. Например, первым пойдет вычитание. Точнее, унарный минус или попросту отрицательные величины: бинарного минуса в *Maxima* по сути не существует, а любое выражение вида **a–b** имеет внутреннюю форму a+(–b), то есть сводится по все тому же принципу к плюсу. Итак, приступим:

```
setup_autoload(stringproc,sequal)$
deriv([l]):=block([f,len,x,o],
...
        else
            return(0),
    o:op(f),
    return (
        if sequal(o,"-") then
            -deriv(-f,x)
        else
            'diff(f,x)
    )
)$
```

Тут мы уже начинаем использовать те самые функции по «глубокой» обработке выражений. Функция op() возвращает основной оператор заданного выражения. Основным считается самый внешний; например op(a+b/c) будет равен "+", op((a+b)*2) – "*", а op(sin(x^2+y^2)) – sin. Дальше включается «принцип чайника»: для отрицательного выражения мы просто выносим минус за скобки, а для остального, теперь уже положительного, вызываем саму же функцию deriv.

Здесь для сверки значения оператора с минусом используется не equal(), а ее строковый аналог – sequal(), проверяющий на равенство две строки. Связано это с тем, что разные операторы *Maxima* хранит в разном виде, и при сверке, скажем, того же минуса, который хранится как текстовый знак с синусом, хранящимся как символ (идентификатор) *Maxima*, обычный equal() просто выдаст ошибку.

Функция sequal() – внешняя, она хранится в файле **stringproc** (от фразы «string processing» – обработка строк), который и нужно подгрузить до использования этой функции. А для того чтобы, файл не приходилось загружать вручную, но при этом он и не загружался бы при каждом вызове функции (как было бы в случае вызова load() внутри функции deriv()), есть, с одной стороны, традиционный способ: определить внутри файла некую константу или свойство, а перед его загрузкой проверять их наличие: если нету – тогда и подгружать. Мы же используем не общепринятый, но в чем-то более простой метод: рассмотренную в статье функцию setup_autoload. Благодаря ей, нам с одной стороны, не надо лезть в исходники библиотек (которые, кстати говоря, часто бывают не на языке *Maxima*, а на Lisp) и искать там флаги; а с другой – мы все же уверены, что файл будет загружаться не больше одного раза: именно это и гарантируется функцией setup_autoload.

И последний момент в этом кусочке: обратите внимание на оператор if, сместившийся внутрь функции return(). Напомню, что if в *Maxima* является полноценным оператором, то есть всегда возвращает последнее вычисленное значение. А раз так, нет никакого смысла вызывать return() много раз. По большому счету, здесь и один вызов return() не нужен: результатом block(), как и примитивного составного оператора, будет последнее вычисленное выражение. Так что для еще большей краткости напишем даже так:

```
...
    o:op(f),
    if sequal(o,"-") then
        -deriv(-f,x)
    else
        'diff(f,x)
)$
```

После минуса логично было бы заняться плюсом; но поскольку сумма при дифференцировании переходит в сумму, то проще будет реализовать ее сразу для произвольного числа слагаемых, а это уже немного сложнее. Потому начнем с более простых в реализации арифметических действий: умножения и деления.







»

»

```
...
    if sequal(o,"-") then
        -deriv(-f,x)
    else if sequal(o,"*") then
        deriv(first(f),x)*rest(f)+first(f)*deriv(rest(f),x)
    else if sequal(o,"//") then
    (deriv(first(f),x)*last(f)-first(f)*deriv(last(f),x))/last(f)^2
    else
        'diff(f,x)
)$
```

Здесь мы сталкиваемся с одним очень интересным и весьма полезным свойством: многие из функций работы со списками, которых в *Maxima* немало, воспринимают как списки также и любые выражения. Так, «списковая» функция first(), возвращающая первый элемент заданного списка, вызванная как first(a*b*c), вернет **a**; а у функции rest() («остаток»), отдающей (в варианте вызова с одним аргументом), наоборот, весь список кроме первого элемента, на том же выражении результатом будет **b*c**. Этим мы и воспользовались, вызывая при этом снова для каждого слагаемого саму функцию deriv(). Если сомножителей будет больше чем два, то вызов deriv(rest(f),x) пройдет по этой же ветке и отсечет еще один.

Так же мы поступаем и с делением. Здесь, так как аргумента всегда два, вместо rest() используется функция last() – последний элемент списка (rest() в этом же случае вернула бы список из одного элемента, а потому last() более удобна). Только одно «но»: деление почему-то обозначается во внутреннем представлении *Maxima* не одиночной, а двойной косой чертой.

Точно таким же образом можно обработать последний бинарный оператор (кроме оставленного на закуску сложения) – возведение в степень. Здесь тоже нет никаких сложностей, и даже нечего дополнительно объяснять по сравнению с делением:

```
...
    else if sequal(o,"^") then
        first(f)^last(f)*log(first(f))*deriv(last(f),x)+
        first(f)^(last(f)-1)*last(f)*deriv(first(f),x)
    else
        'diff(f,x)
)$
```

Теперь вернемся к сложению. Тут нам уже пригодятся упомянутые в статье функции по работе с функциями, а конкретно – функция map(). Она принимает в качестве первого аргумента имя функции и как бы вкладывает эту функцию внутрь выражений – последующих аргументов. Проще всего будет пояснить на примере: map(f,[a,b,c]) даст результат [f(a),f(b),f(c)]. И, что самое замечательное, она, точно так же, как и «списковые» функции, работает не только со списками, но и с любыми выражениями; например, map(f,a+b+c) –> f(a)+f(b)+f(c). Как хорошо подходит для нашей задачи, не правда ли? Именно так и должна действовать на сумму функция дифференцирования. Все было бы совсем хорошо, если бы deriv() принимала, кроме выражения, только один аргумент. С двумя выражениями map тоже умеет работать, но только если у них одинаковый основной оператор; то есть сумму можно «отобразить» только на сумму: map(f,a+b+c,x+y+z) –> f(c,z)+f(b,y)+f(a,x). Проблема здесь в том, что у нас второй аргумент во всех вызовах deriv(), которые должны попасть внутрь суммы, одинаков, а выражение вида x+x+x передать невозможно: оно автоматически упростится в 3*x. Но, как известно, из любой безвыходной ситуации всегда есть как минимум два выхода. И в данном случае один из этих выходов достаточно прост: написать небольшую функцию-«обертку» вокруг map:

```
map1st(f,expr,x):=block([o],
    o:op(expr),
    subst(o,"[",map(f,subst("[",o,expr),makelist(x,i,1,length(expr))))
)$
```

Еще одна новая функция «глубинной» работы с выражениями: subst(). Она способна заменять в выражении... да почти что угодно и почти на что угодно. Вызывается так: subst(стало, было, выражение), заменяя в «выражении» все, что «было», на «стало». Опять же, в качестве подвыражений могут использоваться операторы, то есть subst("*","+",x+y+z) –> x*y*z. Мы используем ее для временной подмены основного оператора выражения оператором списка (который обозначается как "[", то есть [a,b,c] – это, по сути, "["(a,b,c)). Затем генерируем список такой же, как выражение, длины, заполненный заданной переменной, – и применяем к двум полученным спискам функцию map(), а затем возвращаем назад вместо списка первоначальный базовый оператор. То есть теперь, к примеру, map1st(f,a+b+c,x) будет равно как раз f(c,x)+f(b,x)+f(a,x). Et voila, как говорят французы! И теперь внутри deriv() можно применить к сложению именно эту новую функцию. Заодно применим ее и к списку – (deriv([f,g],x) будет равно [deriv(f,x),deriv(g,x)]) и, чего уж там мелочиться, и к множеству:

```
...
    o:op(f),
    if sequal(o,"+") or sequal(o,"[") or sequal(o,set) then
        map1st(deriv,f,x)
    else if sequal(o,"-") then
        -deriv(-f,x)
...
```

Множества, к слову, в *Maxima* реализованы в самом что ни на есть математическом смысле: множество может включать в себя каждый элемент только один раз; и это учитывается и встроенными операциями по работе с множествами: пересечением, объединением и т.д. Есть еще некоторые ошибки, но они документированы и потому не неожиданны.

Движемся дальше. У нас уже реализована производная от всех бинарных операторов, а дальше мы нарисуем «таблицу производных» и будем работать с нею:

```
deriv([l]):=block([f,len,o,x,func,fdrv],
...
    o:op(f),
    func:[sqrt, sin, cos, abs, exp, log, tan, cot, sec, csc, asin, acos, atan,
acot, asec, acsc, sinh, cosh, tanh, coth, asinh, acosh, atanh, acoth,
asech, acsch],
    fdrv:[1/2/arg, cos(arg), -sin(arg), arg/abs(arg), exp, 1/arg, sec(arg)^2,
-csc(arg)^2, tan(arg)*sec(arg), -cot(arg)*csc(arg), 1/sqrt(1-arg^2), -1/
sqrt(1-arg^2), 1/(1+arg^2), -1/(1+arg^2), 1/arg^2/sqrt(1-1/arg^2), -1/
arg^2/sqrt(1-1/arg^2), cosh(arg), sinh(arg), sech(arg), -csch(arg), 1/
sqrt(arg^2+1), 1/sqrt(arg^2-1), 1/(1-arg^2), 1/(1-arg^2), -1/arg^2/sqrt(1/
arg^2-1), -1/arg^2/sqrt(1/arg^2+1)],
    if sequal(o,"+") or sequal(o,"[") or sequal(o,set) then
...
```

Для упрощения работы с «таблицей» напишем еще две небольших вспомогательных функции: одна будет проверять, входит ли заданный элемент в заданный список, а вторая – возвращать номер, соответствующий заданному элементу в заданном списке, при условии что он там есть.

```
smember(expr,list):=
    if sequal(true,
        for i in list do
            if sequal(expr,i) then
                return(true) )
    then true$
sindex(expr,list):=block([num],
```



```
    num:for i:1 thru length(list) do
            if sequal(expr,list[i]) then
                return(i),
    if integerp(num) then num
)$
```

Здесь есть только одна тонкость, связанная с небольшой проблемой. Заключается эта проблема в том, что для возвращения значения из блока и из цикла в *Maxima* используется одна и та же функция return(). Это приводит к тому, что выйти из блока, находясь внутри цикла в нем, невозможно – приходится выдумывать некоторые несложные ухищрения. Теперь с использованием двух новых функций заменяем элементы «таблицы» их производными; с помощью уже знакомой нам subst, которая подставит нужное выражение внутрь табличной функции вместо ключевого слова arg.

```
...
    else if smember(o,func) then
    deriv(first(f),x)*subst(first(f),arg,fdrv[sindex(o,func)])
    else
        'diff(f,x)
)$
```

Вот так, начиная с самых простых элементов, а затем, подобно Мюнхгаузену, вытаскивая самих себя сантиметр за сантиметром, мы и получили полноценную функцию дифференцирования. Правда, пока только первого порядка и только по одному аргументу. Но имея то, что имеем, двигаться дальше, следуя известному принципу, уже совсем не сложно: просто заменим строку «if len>=3 then error...» следующим куском:

```
...
    if len=3 then (
        integerp(l[3]) or
            return('diff(x,l[3])),
        if l[3]=0 then
            return(f),
        if l[3]<0 then
            error("Improper count to deriv:",l[3]),
        if l[3]>1 then
            return(deriv(deriv(f,x,l[3]-1),x))
    ),
    if len>3 then (
        if(evenp(len)) then
            l:endcons(1,l),
        return(deriv(apply(deriv,rest(l,-2)),l[len],l[len+1]))
    ),
...
```

Пройдемся по нескольким неосвещенным моментам. В силу способов вычисления в *Maxima* (которые сродны таковым во многих языках программирования) конструкция вида «условие or выражение» равносильно «if not условие then выражение» – и использована здесь исключительно для разнообразия, в учебных целях. Здесь мы в случае нецелого порядка дифференцирования просто возвращаем несовершенную форму – точно так же, как это делает и штатная функция diff().

Производная нулевого порядка от любой функции – это сама функция. А производные отрицательных порядков некорректны, о чем мы и генерируем сообщение. Для порядков, больших единицы, понижаем порядок как и раньше – за счет самовызова.

Далее я немного усовершенствовал поведение функции по сравнению со встроенной: если та не умеет принимать четное количество аргументов больше двух (то есть с неуказанным порядком дифференцирования по последней неизвестной когда неизвестных больше одной), то у нас в данном случае, так же как и для одной неизвестной, будет подразумеваться единица. Здесь предикат evenp() проверяет число на четность (even – четный), а функция endcons() добавляет заданный элемент в конец заданного списка. Ее имя носит исторический характер: парная к ней функция cons(), добавляющая элемент в начало списка, свое имя позаимствовала из Lisp, а слово end здесь добавлено «по смыслу».

Далее мы, снова самовызовом, укорачиваем список параметров дифференцирования. При этом используется еще одна функция, работающая с функциями, – apply() (применять). Она принимает два аргумента, первый из которых – имя функции, а второй – список, и применяет заданную функцию к списку как к списку аргументов. Также здесь использован более широкий вариант вызова rest(): он может принимать второй аргумент – целое число, не равное нулю. Если число положительно, то такое количество элементов выбрасывается из начала списка, а если отрицательно – то с конца; в данном случае мы теряем последние два элемента.

Вот и все. Мы уже имеем полную функцию дифференцирования, берущую производные с произвольным количеством параметров и любых порядков. Полный текст всех созданных функций вы можете найти в файле deriv.mac на прилагаемом к журналу диске.

Дополнительно хочется остановиться на одной незамысловатой функции, которая, тем не менее, может неплохо помочь в отладке собственных модулей. Это функция display(), которая принимает имена и отображает их значения в виде «имя=значение». В качестве эксперимента можете добавить ее где-нибудь внутри функции deriv() и отследить процесс самовызова (в файле на диске вызов display() достаточно раскомментировать).

И в качестве финального аккорда сделаем еще и более универсальную версию вспомагательной функции map1st() – возможно, тогда она вам пригодится и еще где-нибудь.

```
mapany(f,[lst]):=block([o,l],l:lst,
    if length(setify(map(length,l)))>1 then
    error("Arguments to mapany are not of the same length"),
    o:op(l[1]),
    for i:1 thru length(l) do
        l[i]:subst("[",op(l[i]),l[i]),
    subst(o,"[",apply(map,cons(f,l)))
)$
```

Здесь я уже воздержусь от столь подробных комментариев, так как практически все, что используется в этой функции, уже было в той или иной степени разъяснено в процессе описания deriv(). Остановлюсь только на одной строчке:

```
            if length(setify(map(length,l)))>1 then
```

Здесь используется не совсем простой прием для проверки длин списков на одинаковость. Так как l – это список из списков, то сначала получаем список длин «вкручиванием» внутрь внешнего списка функции length(). Дальше – интереснее. Функция setify (дословно – что-то вроде «множествицировать») превращает список в множество. Так как множество не может содержать несколько равных между собой элементов, то такие элементы при этом «склеиваются»: из них остается один. Таким образом если «длина» (количество элементов) множества больше единицы, то как минимум два элемента в первоначальном списке были неравны между собой.

И вернувшись к рассмотренной функции дифференцирования, хочется еще раз обратить ваше внимание на использованный прием: конструировать большие и сложные функции из более маленьких и простых кусочков с помощью рекурсии. Этот метод очень часто и продуктивно используется в функциональном программировании, к которому *Maxima*, в силу своих Lisp-овских корней, очень близка. LXF











Есть вопрос по Open Source? Пишите нам по адресу: answers@linuxformat.ru

В этом месяце мы отвечаем на вопросы по:

1 Разрешению экрана
2 kded
3 Debian
4 chmod
5 Перезагрузке
6 Множеству разделов
7 KMyMoney
8 WebDAV
9 Привилегиям администратора
10 Open Media Streaming Project
11 Потерянной Windows
12 NFS
13 Неизвестной web-камере
★ NIC
★ Клонам ASP Linux

Значения vga для различных видеорежимов

Глубина цвета \ Разрешение	640x480	800x600	1024x768	1280x1024	1152x864	1600x1200
8-бит	69	771	773	775	353	800
15-бит	784	787	790	793	354	801
16-бит	785	788	791	794	355	802
24-бит	786	789	792	795	–	803

Таблица приводится по материалам и с разрешения http://gentoo-wiki.com/HOWTO_fbsplash.

1 Морока с меню Mepis

Недавно попробовал Mepis с DVD номера LXF84, но там не сказано, что меню загрузки переключаются клавишами F1, F2 и F3. F3 – выпадающее меню для выбора разрешения экрана – в моем случае это 1600x1200. Можно (но я не знаю, как) изменить его после установки, командной строкой, но ведь это сложно для начинающих..

Джим [Jim], с форума LXF

Наш DVD содержит SimplyMEPIS и Knoppix, на выбор, поэтому специфичное для Mepis меню пришлось опустить. Стандартный CD Mepis имеет графическое меню загрузки, но какой видеорежим там выбран, неважно – система всегда будет загружаться в 1024x768, потому что это разрешение зашито в настройках загрузки Mepis 6.0. К счастью, Mepis использует *Grub*, и можно менять опции загрузки на лету.

› Смена разрешения VGA в SimplyMepis. Для нашего DVD,дважды нажмите 'e'.

Обычный способ изменить настройки загрузки – выделить нужный пункт меню и дважды нажать e, для редактирования первой строки. Измените число после 'vga=' на соответствующее нужному видеорежиму. Нажмите Enter для сохранения изменений и b для загрузки с новыми настройками. Пользователи оригинального диска Mepis могут также поменять видеорежим, выделив пункт меню и клавишами-стрелками добравшись до значения VGA в поле ввода снизу. Измените номер аналогичным образом и нажмите Enter. В таблице вверху приведены числовые значения для наиболее часто используемых видеорежимов. Например, если Вы хотите установить разрешение 1600x1200 при глубине цвета в 16 бит, Вам нужно указать vga=802. **НБ**

Наши эксперты

» Мы найдем эксперта на любой вопрос! Вы получите ответ на все: от проблем с установкой или модемом до сетевого администрирования; главное – спросить!



Нейл Ботвик
Владелец ISP и экс-редактор дисков для нашего журнала, Нейл считает, что в Linux он от скуки на все руки.



Майк Сондерс
Майк был одним из создателей прототипа LXF – Linux Answers. Его специальности – программирование, оконные менеджеры, скрипты инициализации и SNES.



Кингс Кобблер
Кингс – системный инженер Linux и администратор Rackspace, использует Linux десять лет, всегда готов отвечать на письма других администраторов.



Ник Вейч
В свободное от исчеркивания текстов красными чернилами время Ник возится с Linux-графикой и 3D-приложениями; он у нас отвечает за простые вопросы!



Андрей Маркелов
Сертифицированный специалист Red Hat (RHCE/RHCI), всегда готов помочь вам с этим дистрибутивом и его производными.



Валентин Синицын
В свободное от работы время редактор нашего журнала разрабатывает KNetworkManager и другие открытые приложения. Он с радостью поможет вам в вопросах использования Linux на рабочем столе.

КУДА ПОСЫЛАТЬ ВОПРОСЫ:
Пишите на м по адресу: answers@linuxformat.ru или спрашивайте на форуме: www.linuxforum.ru

2 Kded'овщина

Недавно установил Mandriva 2006 Free с диска к LXF75. При каждой загрузке машины, через несколько минут все начинает так тормозить, что невозможно работать. Мой *GKrellM* показывает, что процессор полностью загружен. Проверка с помощью *top* указывает на *kded*. Когда я его убиваю, проблема исчезает. Что делает этот демон и можно ли его отключить вообще?

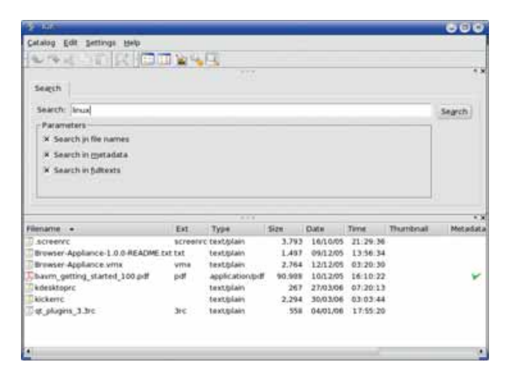
mikedj, с форума LXF

О У Вас установлена утилита поиска *Kat*? Тогда, вероятно, это и есть главный злодей. *Kat* вызывает *kded* в процессе своей работы, и хотя в top просматривается *kded*, проблема кроется в *Kat*, способной снизить производительность мощных машин до уровня ZX81. Впрочем, новейшие версии менее требовательны. *Kded* – сервисный демон среды KDE. Он обрабатывает обновления *Sysoca*, базы данных с информацией о приложениях. Возможно, он съедает часть циклов Вашего процессора.

Кардинальное решение – удалить *Kat*; но можно просто убить его процессы:

```
killall kat
killall katdaemon
killall kded
```

Первые две строки относятся к *Kat*; третья убивает *kded*, который все еще пытается обработать запросы *Kat*. Заглянув в список процессов, Вы обнаружите, что KDE перезапустил *kded*, но он больше не тормозит работу Вашей системы.



› ***Kat* способен сильно затормозить вашу систему, но его можно отключить.**

Чтобы предотвратить запуск *Kat* после перезагрузки, выполните команду:

```
touch ~/.mdv-no_kat
```

Если Вам понадобится вновь активировать *Kat*, просто удалите файл **~/.mdv-no_kat**. НБ

3 Apt-get не обновляет

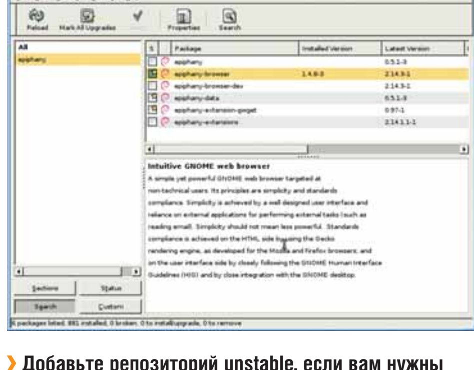
В Пытаюсь обновить web-браузер *epiphany* на Debian 3.1 Sarge, используя

```
su && apt-get install epiphany-browser
```

И вот что получается:

```
Reading Package Lists... Done
Building Dependency Tree... Done
epiphany-browser is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
2 not fully installed or removed.
Need to get 0B of archives.
After unpacking 0B of additional disk space will be used.
Setting up kernel-image-2.4.27-3-686 (2.4.27-10sarge3)
...
cp: writing `/tmp/mkinitrd.MMwVww/initrd//lib/
libc.so.6': No space left on device
cp: writing `/tmp/mkinitrd.MMwVww/initrd//lib/
libe2p.so.2': No space left on device
run-parts: /usr/share/initrd-tools/scripts/e2fsprogs
exited with return code 1
Failed to create initrd image.
```

Хм, epiphany-browser уже обновлен до новейшей версии? Я видел, что *Epiphany 1.7.6* уже доступен для загрузки. Может, все дело в медленном обновлении Debian и 1.4.8 – действительно последняя версия в репозиториях Debian...



› **Добавьте репозиторий unstable, если вам нужны новые версии программ.**

И что это за «нет места на диске»? На всем жестком диске занято около 14%.

А.М.С. Брэдли [AMS Bradley]

О Последний релиз *Epiphany* – 2.14.3, из ветви 1.x – 1.8.5. В стабильной ветви Debian 1.4.8 действительно последняя версия, однако в Testing доступны 1.8.5 и 2.14.3. Пакеты, доступные в каждой ветви Debian, можно посмотреть тут: http://packages.debian.org. Вам нужно добавить репозитарий testing в Ваш **/etc/apt/sources.list** вручную или при помощи *Synaptic*. Проще всего это сделать, скопировав строку со stable и заменив stable на unstable.

Что касается сообщения о нехватке места – это относится к директории **/tmp**. Она расположена на отдельном разделе? Это хорошая идея и наиболее частый способ предотвратить переполнение диска, если какой-нибудь процесс заполнит все своим временным файлом. Я подозреваю, что так и было, в результате раздел с **/tmp** переполнился. Попробуйте ввести

```
df -h
```

Если раздел **/tmp** полностью забит, то причина в этом. Можете удалить из этой директории любой файл, который был создан до последней перезагрузки. МС

4 Биты chmod

В Я новичок в Linux и немного запутался, как работают биты прав доступа в *chmod*. Можете ли вы мне помочь?

Азим Мохаммед [Asim Mohammed]

О Права доступа в *chmod* выражаются тремя восьмеричными цифрами либо группами символов. Это трио представляет права для владельца файла, группы владельца и всех остальных. Возьмем *chmod* 755. Каждая цифра – это сумма, сложенная из различных привилегий. Вот их значения:

```
0 – нет прав (---)
1 – только выполнение (--x)
2 – только запись (-w-)
3 – запись и выполнение (-wx)
4 – только чтение (r--)
```

Для установки прав на чтение и выполнение нужна цифра 5: 1 для выполнения и 4 для чтения. Для полного доступа нужно 4 для чтения, 2 для записи и 1 для исполнения: 4 + 2 + 1 = 7.

Если Вы устанавливаете на файл права 755, это значит, что владелец имеет полный доступ (7), группа имеет право читать и выполнять файл (5). Точно такие же права имеют все остальные. Есть еще несколько битов для специфичных функций, но эти – основные. ДК »



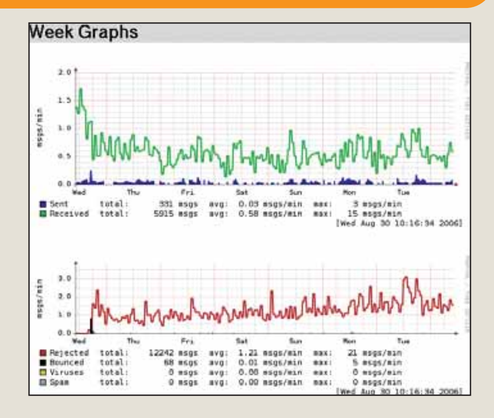
Краткая справка

Серые листы

Вооружитесь перед схваткой со спамом подобной утилитой.

Спам, спам, спам. Это пожиратель времени, трафика и ресурсов системы. Мы пытаемся сократить эти потери, фильтруя почту и помечая спам-письма. Чем лучше это делается, тем сложнее спамерам обходить фильтры. Байесовские фильтры вроде *SpamAssassin* проверяют содержимое каждого письма; это эффективно, но сильно растрачивает системные ресурсы.

Серые списки предлагают другой подход к фильтрации, еще до анализа контента. Когда новое письмо принимается от незнакомого адресата или отправляется человеку, которому вы еще не писали, фильтр отсеивает его со стандартным кодом ошибки SMTP 451, означающим «сервер временно недоступен, пожалуйста, попробуйте позже». Любое стандартное почтовое ПО повторит попытку через несколько минут. Когда то же письмо приходит снова, ПО фильтрации по серым спискам принимает его и вносит в базу данных, чтобы следующие письма принимались сразу. Почему это работает? Потому что ПО для рассылки спама использует метод «отошли и забудь», чтобы отправить как можно больше сообщений, и не обращает внимание на ошибки и недоставленные письма. Ошибка будет проигнорирована, и сервер не повторит попытку, а через день или около того отправитель будет включен в черный список и все его будущие письма будут удаляться. Это решение снижает нагрузку на почтовый сервер, ценой небольшой задержки доставки первого письма.



› **Посмотрите, как много мусора ликвидировано без помощи спам-фильтра.**

Функция фильтрации по серым спискам доступна для большинства популярных почтовых серверов. Например, для Postfix есть *Postgrey*, по адресу http://isg.ee.ethz.ch/tools/postgrey.

Подробности на www.greylisting.org.

5 Семейный SUSE

В У меня есть компьютер с SUSE 10.0, им пользуются все в моем доме для web, почты, просмотра ТВ и фильмов и прослушивания музыки. Никто в моей семье не имеет пользовательского опыта в Linux, и когда меня нету рядом, начинаются ссоры. Компьютер подключен к высокоскоростному каналу по Wi-Fi, который нормально работает большую часть времени. Сложности бывают, когда у моего провайдера случаются технические проблемы и меня выбивает из сети (обычно в субботу утром), после чего нужно перезапустить беспроводной маршрутизатор и сеть на компьютере. Для меня-то это не проблема, но я не могу объяснить своей семье, зачем вообще нужен терминал и зачем все эти шаманства с root. Им подавай иконку, вызывающую перезапуск сети. Можете помочь?

Дейв Вайз [Dave Wise]

О Если Ваши домочадцы не понимают, зачем нужен пароль root, ни в коем случае им его не давайте! Именно в таких ситуациях нужен *sudo*, описанный в «Кратком руководстве» в прошлом номере. Вам нужно создать скрипт, содержащий последовательность команд для перезапуска сети и назвать его, скажем, **/usr/local/bin/restartnetwork**. Убедитесь, что владельцем файла будет root, и что только он будет иметь права на запись в файл:

```
chown root: /usr/local/bin/restartnetwork
chmod 755 /usr/local/bin/restartnetwork
```

Добавьте следующую строку в **/etc/sudoers**, чтобы позволить всем членам группы users выполнять скрипт без пароля:

```
%users ALL = NOPASSWD: /usr/local/bin/restartnetwork
```

Это позволит им выполнять перезапуск сети без знания пароля root. Если Вы поменяете NOPASSWD на PASSWD, пользователь должен будет ввести собственный пароль. Вместо целой группы Вы можете указать отдельных пользователей, через запятую:

```
ma,pa,johnboy ALL = NOPASSWD: /usr/local/bin/restartnetwork
```

Теперь авторизованные пользователи могут выполнять скрипт так:

```
sudo restartnetwork
```

Полный путь к скрипту не нужен, если **/usr/local/bin** находится в переменной окружения $PATH, однако его нужно обязательно указывать в **/etc/sudoers**. Создав скрипт, привяжите его к иконке на рабочем столе или кнопке на панели, и любой из Ваших пользователей сможет перезапустить сеть одним щелчком мыши. Поскольку *sudo* запускает скрипт с правами root, то и все команды скрипта будут выполняться с привилегиями суперпользователя, хотя Ваша семья не имеет к ним доступа.

Таким методом я добавил кнопку запуска беспроводной сети на панель на своем ноутбуке. Не потому, что я не знаю пароля root, а потому, что я ленив, и лучше уж один клик, чем ввод пароля. НБ

6 Разделяй и властвуй

В Мой корневой раздел почти заполнен. Мне нужно больше места, и у меня есть свободный раздел, куда я могу положить, скажем, **/usr/lib**. Но как это сделать?

jensjk, с форума LXF

О Linux позволяет монтировать отдельные разделы где угодно под **/**, упрощая использование отдельных разделов для каждой части Вашей системы. Самая хитрая часть процесса – перенос данных из оригинальной файловой системы на новую. Если Вы не используете отдельный раздел под **/home**, я очень рекомендую сделать это, поскольку отделение **/home** от остальной файловой системы имеет несколько преимуществ. Что бы Вы ни делали, сделайте резервные копии. Если Вы удалите что-то нужное, то будете рады, что послушались меня. Копирование данных, особенно системных файлов, на живой системе – рискованное занятие, поэтому загрузитесь с LiveCD, например, с KNOPPIX. Я подразумеваю, что Ваш текущий системный раздел – **/dev/hda1**, а **/home** Вы будете перемещать на **/dev/hda2**. Внесите соответствующие изменения, в соответствии с Вашей системой.

Первый шаг – запуск *QtParted*, разметка и форматирование раздела. Теперь откройте терминал и выполните следующие команды:

```
su
mount /dev/hda1 /mnt/hda1
mount /dev/hda2 /mnt/hda2
rsync -avx /mnt/hda1/home/ /mnt/hda2/
```

Первая строка дает привилегии суперпользователя, следующие две монтируют старый и новый разделы, последняя – копирует все из старой директории home в новый раздел. Можно было бы использовать cp или tar для копирования файлов, но по-моему, самый надежный метод создания копии – *rsync*, если нужно сохранить временные отметки и права доступа.

Теперь добавьте строку, относящуюся к новому разделу, в **/etc/fstab**. Knoppix поставляется с редактором *Nano*:

```
nano /mnt/hda1/etc/fstab
```

Добавьте что-то вроде

```
/dev/hda2 /home ext3 defaults 0 0
```

Это если Вы отформатировали раздел в файловой системе *ext3*, как это делает по умолчанию *QtParted*. Если Вы используете *ReiserFS*, замените **ext3** на **reiserfs**.

Если Вы загрузитесь в систему и выполните

```
df -h
```

в терминале, то увидите, что **/home** (или любая другая директория, которую Вы переносили на другой раздел), теперь находится на отдельном разделе. «Но», вскрикнете Вы в монитор, «мой корневой раздел все еще заполнен!»

Это потому, что Вы скопировали данные на другой раздел, и они теперь в находятся в обоих местах. Так »



Часто задаваемые вопросы…

Магические клавиши

Контролируйте ядро для безопасной перезагрузки.

» Зачем на ПК есть кнопка перезагрузки?
Потому что они поставляются с ОС от Microsoft.

» Старая шутка, еще времен динозавров. По-вашему, Linux-программы никогда не падают?
Ну, ядро Linux очень стабильно, это правда, и падение приложения обычно не влияет на систему в целом. Но какой-нибудь процесс может захватить слишком много процессорного времени и оперативной памяти, и потребуется перезагрузка. Или ядро запаникует и прервет загрузку системы.

» Значит, кнопка перезагрузки все-таки нужна?
Нет, не нужна. У Linux есть немного магии в ядре: нажмите Alt, SysRq и клавишу команды, и ядро обязательно отзовется, если оно, конечно, полностью не заблокировано. Поскольку это функция ядра, она становится доступна сразу после его загрузки, и ее можно использовать при инициализации системы.

» У меня нет кнопки SysRq. Что мне делать?
SysRq используется так же, как PrintScreen – она может быть помечена одним из этих названий или обоими.

» Что такое командные клавиши?
Есть несколько командных клавиш для выполнения различных операций, но самые полезные – S, U и B, в порядке перечисления. S синхронизирует все смонтированные файловые системы; это значит, что содержимое дискового кэша будет немедленно сброшено на диски. U размонтирует все смонтированные файловые системы и переподключит их в режиме только для чтения. Эти две операции дают уверенность в том, что содержимое дисков не пострадает.

После этого можно безопасно выполнить третью команду B для перезагрузки (перезагрузка происходит немедленно). Поскольку диски синхронизированы, после перезагрузки не будет никаких предупреждений, и нет нужды запускать fsck.

» Это хорошо при доступе к клавиатуре компьютера, а если я подключен удаленно?
Вы можете выполнить эти магические команды так:

```
echo s >/proc/sysrq-trigger
```

При удаленном доступе к машине, можно выполнить синхронизацию и завершение работы.

» Как запретить другим использование этих клавиш?
Строки

```
echo 0 >/proc/sys/kernel/sysrq
echo 1 >/proc/sys/kernel/sysrq
```

включают и выключают эту функцию соответственно; это разрешается только root. Можно также запретить отдельные команды, это описано в документации.

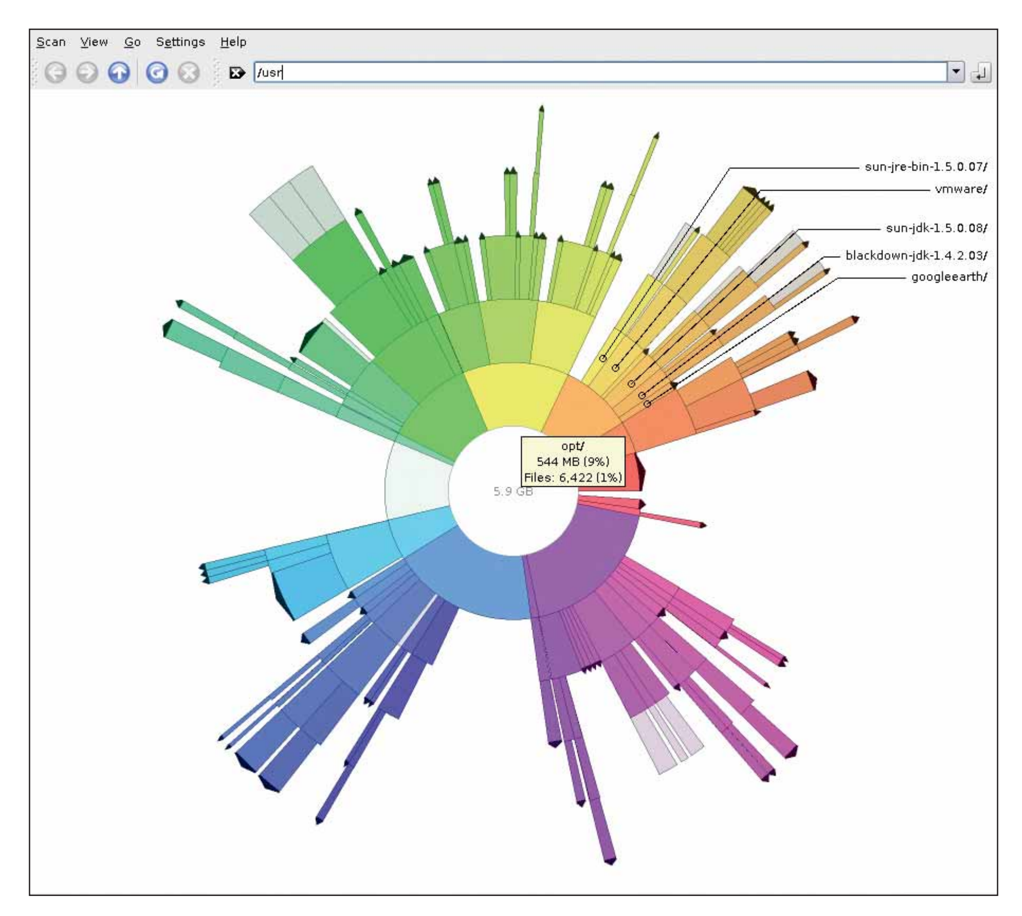
» Какие другие клавиши и команды я могу использовать?
Их слишком много, чтобы все их здесь описать; детали можно почерпнуть из документации к ядру. Если установлены исходные коды ядра, то она в файле **/usr/src/linux/Documentation/sysrq.txt**. В противном случае можете посмотреть их в сети по адресу www.gelato.unsw.edu.au/lxr/source/Documentation/sysrq.txt. Удачи!

» и задуманО в случае неверных действий Вы сможете все откатить. Данные все еще здесь, но они невидимы, поскольку новый раздел смонтирован в **/home**, и старые данные скрыты. Можете загрузиться в Knoppix и удалить данные оттуда, как только убедитесь, что все в порядке, но вот небольшой трюк для избежания перезагрузки:

```
mkdir /mnt/tmp
mount --bind / /mnt/tmp
rm -fr /mnt/tmp/home/*
```

Он позволит Вам увидеть и удалить файлы в старой домашней директории. Убедитесь, что Вы удалили только содержимое, а не саму директорию. Она нужна, чтобы смонтировать новый раздел. Можете сделать то же самое с **/usr/lib**, если захотите, однако **/home** – лучший вариант, если он еще не на отдельном разделе (в противном случае обдумайте перемещение **/usr/local**). Все зависит от того, сколько места Вы хотите освободить; поможет информация о том, сколько места занимает каждая директория, от утилиты *FileLight*, доступной на www.methylblue.com/filelight и включенной в репозитории многих дистрибутивов.

Можете также использовать *LVM* для комбинирования дисков, как описано в LXF83. **НБ**



› ***FileLight* – простой способ узнать, куда девалось место на диске. И на вид мило.**

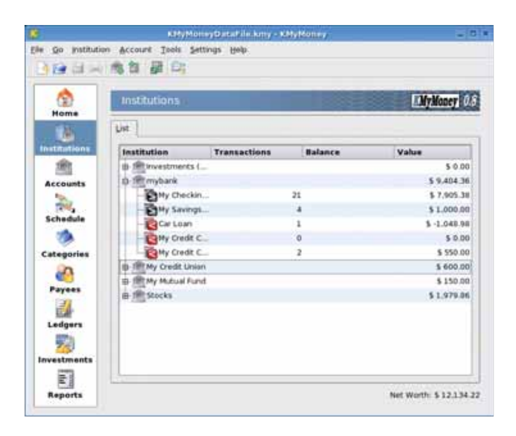
7 Перевод денег

В Я использую Linux уже около трех лет; два года у меня был Mandrake и год – Ubuntu. Я пытаюсь сагитировать и мою половину – она уже использует *OpenOffice.org* на своей машине с XP. Я загрузил для нее Ubuntu Dapper Drake, но есть две сложности. Первая сложность – я не могу экспортировать файлы *MS Money* в *KMyMoney*: данные экспортируются в QIF, но *KMyMoney* его не распознает; возможно, это Microsoft-версия формата QIF.

Другая проблема – она пользуется *MS AutoRoute*, а в Linux я эквивалента не нашел. Я думал об использовании *Wine* как альтернативы, но я ничего не знаю о нем, и не знаю, как установить в нем программу от Microsoft. Буду благодарен за любую помощь.

Джон Мортон [John Morton]

О Ответ на Ваш первый вопрос – используйте другую программу конвертации файлов. *GnuCash* импортирует Microsoft QIF, который Вы можете сохранить в собственный формат *GnuCash*. *GnuCash* имеет опции для обработки нескольких вариаций формата QIF (я успешно импортировал файлы из MS Money). *KMyMoney* имеет опцию импорта файлов *GnuCash* – я ею пользуюсь потому, что храню свои данные в *GnuCash*, но мне нравятся опции отчетов в *KMyMoney*. Формат *GnuCash* хорош тем, что он неизменен, а QIF-файлы могут немного различаться. Вы можете также обновить версию *KMyMoney* – уже заявлено об улучшении поддержки QIF.



› **Импорт QIF прямо в *KMyMoney* имеет сложности, так что попробуйте *GnuCash*.**

Пакет планирования маршрута для Linux – *Navigator* (www.directions.ltd.uk). Это коммерческий продукт, работающий на x86 в Windows и Linux. Демо-версия отсутствует, так что проверьте список совместимости оборудования. Установка *Wine* в Ubuntu очень проста, пакет есть в репозитарии Universe. Выберите Настройки > Репозитории в *Synaptic* и поставьте галочку напротив `Ubuntu 6.06 LTS (binary) Community maintained (Universe)', закройте окно репозитариев и нажмите Обновить (Reload). По завершении процесса, используйте функцию поиска (Search) для нахождения *Wine*. Когда *Synaptic* установит пакет, скомандуйте **winecfg** для настройки, однако в большинстве случаев подойдут настройки по умолчанию.

Теперь можно запустить Windows-программу так:

```
wine /path/to/someprogram.exe
```

Энди Ченнел написал расширенное руководство по *Wine* для начинающих на стр. 46. Попробуйте посмотреть базу данных *Wine* http://appdb.Winehq.org для получения информации о поддержке различных приложений. Можете рассмотреть также *CrossOver* Linux, коммерческую производную от *Wine* (www.codeweavers.com). **НБ**

8 WebDAW-демон

В У меня WebDAV-доступ к учетной записи на Hotmail. А можно ли получить почту на мой POP3-сервер?

Хэриш Тэндон [Harish Tandon]

О Есть несколько инструментов для выполнения этой работы. Я предпочитаю *Hotwayd*. Он запускается как обычный сервис *inetd* и может работать в связке с *Fetchmail*. Скачайте исходные тексты с http://hotwayd.sourceforge.net и установите его с Вашими настройками. Когда закончите, нужно будет активировать его. Чтобы сделать это с помощью *xinetd*, создайте в директории **/etc/xinetd.d** файл **hotwayd** с таким содержимым:

```
service hotwayd
{
        only_from = 127.0.0.1
        disable = no
        type = unlisted
        socket_type = stream
        protocol = tcp
        wait = no
        user = nobody
        groups = yes
        server = /usr/sbin/hotwayd
        port = 1100
}
```

Теперь просто перезапустите xinetd, и можете использовать *Fetchmail*. Просто создайте в своей домашней директории файл **.fetchmailrc** и запишите в него

```
poll localhost protocol pop3 port 1100 username
"пользователь@сервер.com" password
"ваш_пароль"
```

Теперь запустите *Fetchmail*. Он скачает почту с *Hotmail* на Ваш локальный сервер. **ДК**

9 Зачем же ты так, админ?

В Я сдуру удалил административные привилегии у всех трех пользователей на своей Ubuntu. Это укоротило меню «Администрирование» в Gnome до нескольких пунктов. В данный момент у меня нет ни одного пользователя в группе *admin*, под которым можно администрировать систему. Я могу использовать Терминал и *sudo*, но не могу перевести ни одного пользователя обратно в администраторскую группу. Пробовал команду **gpasswd -g**, но не могу заставить ее верно работать. Как мне добавить моего пользователя *master* снова в администраторскую группу?

ellip, с форума LXF

О Во-первых, не беспокойтесь о том, что совершили ошибку, мы все их делаем. На них-то мы и учимся, главное – никому про них не рассказывать. Команда для добавления Вашего пользователя в группу *admin* выглядит так:

```
gpasswd -a master admin
```

Однако только root имеет право манипулировать базами данных паролей и групп, так что Вы оказались в ловушке: чтобы выполнить **gpasswd** для добавления себя в администраторскую группу, нужно уже быть в этой группе.

Не отчаивайтесь, есть простое решение. Инсталляционный диск является также LiveCD, позволяющим запускать команды от root с помощью *sudo*. Загрузитесь с диска, откройте терминал и запустите

```
sudo bash
mount /dev/hdaN /mnt
nano /mnt/etc/group
```

Замените **hdaN** на раздел, где установлена Ubuntu. Nano – это простой в использовании консольный текстовый редактор. Листайте ниже до строки, начинающейся с **admin:x:112:** и добавьте в ее конец master, чтобы она приняла такой вид:

```
admin:x:112:master
```

Вы можете добавить несколько пользователей, через запятую:

```
admin:x:112:master,slave
```

Не волнуйтесь, если номер не **112:** оставьте его как есть. Нажмите Ctrl-X для сохранения и выхода, а затем загрузитесь с жесткого диска. Теперь у Вас опять должны быть администраторские привилегии. **НБ**

10 Linux для Video

В Можно ли приспособить Linux для создания ТВ-станции на web, транслирующей в Интернет смесь живого видео с web- и видеокамер? Если да, то можно ли это сделать полностью на Open Source?

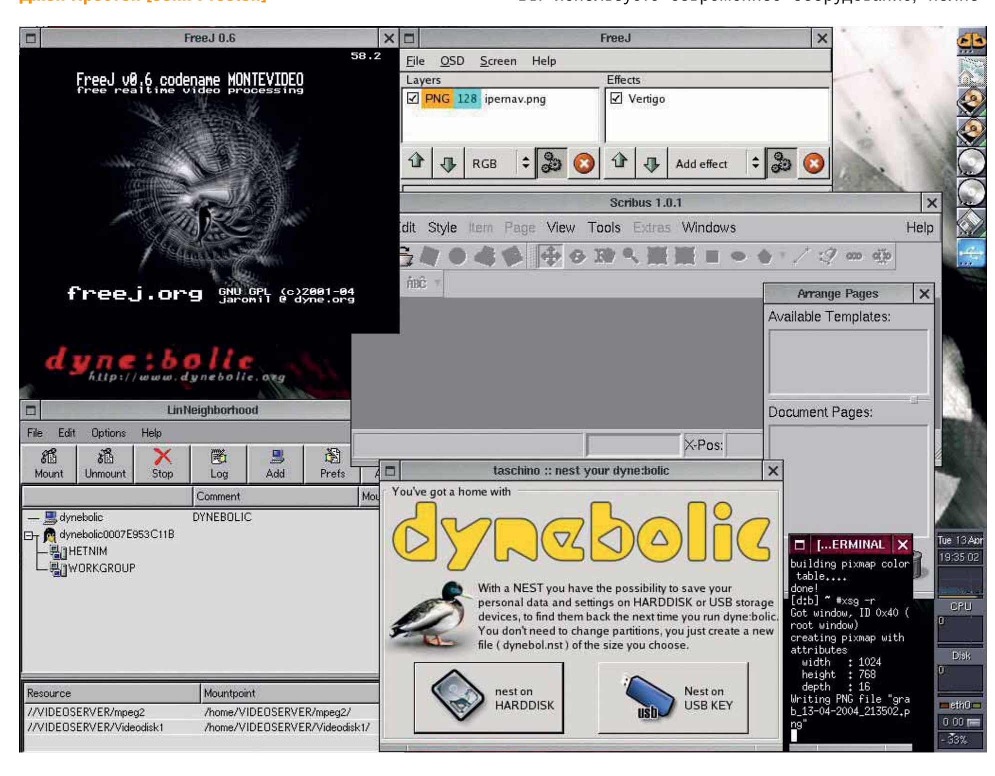
Джон Престон [John Preston]

О Да, все это можно. Вы не сообщили нам подробностей о намеченном Вами проекте, так что дать конкретные рекомендации затруднительно, но (LS)3 Open Media Streaming Project выглядит подходящей отправной точкой. Он включает мультимедийный потоковый сервер *Fenice* и множество полезной документации, в том числе о вещании с живых видеопотоков. *Fenice* поддерживает Video4Linux-устройства, так что подойдет любая работающая с Linux web-камера. Сайт проекта (http://streaming.polito.it) содержит форум, где можно детально обсудить информацию с пользователями и разработчиками.

Другой заслуживающий внимания сервер – *Flumotion* (www.fluendo.com). Это коммерческий продукт, однако базовая часть доступна бесплатно, под GPL. Для смешивания изображения и наложения эффектов в реальном времени можно воспользоваться FreeJ (http://freej.dyne.org).

Наконец, поинтересуйтесь Dynebolic, дистрибутивом для создания и вещания мультимедийного контента. Он может работать как LiveCD, позволяя попробовать свой функционал перед установкой. Последнюю версию можно получить на www.dynebolic.org.

Удачи Вам с Вашим проектом, и дайте нам знать, когда он станет общедоступным! **НБ**



› **Dynebolic имеет множество инструментов для обработки и вещания видео.**

Просто NIC

В Мой компьютер имеет очень низкую производительность сети. Кто-то мне говорил, что проблема, возможно, в полудуплексном режиме (что бы это ни было). Как мне это узнать и заодно посмотреть скорость соединения?

Джин-Гай Леконт [Jean-Guy Leconte]

О Сначала объясню про полудуплекс. Вкратце, это означает, что Ваша сетевая карта договорилась с Вашим сетевым оборудованием и посылает и отправляет пакеты не одновременно; в сущности это вроде одностороннего разговора. Если Вы используете современное оборудование, полнодуплексный режим должен включаться без проблем. Когда NIC соединяется с сетевым оборудованием, он ведет переговоры о скорости и настройках дуплекса на физическом уровне. На большинстве дешевых коммутаторов это делается автоматически: switch объявляет, какие режимы он поддерживает, NIC выбирает один из них и сообщает об этом. Это поведение по умолчанию для большинства NIC.

На оборудовании подороже эти настройки можно зафиксировать для получения оптимальной производительности. Бывает, что на стороне коммутатора все зафиксировано, но Ваша машина установлена на автоматический режим, и это вызывает несовпадение режимов дуплекса, снижающее производительность. Чтобы узнать, в каком режиме работает Ваш NIC, используйте команду **ethtool**:

```
[root@dan ~]# ethtool eth0
```

Выведется детальная информация. Обратите внимание на параметры **Duplex** и **Speed**; Вы также увидите, какие режимы поддерживает коммутатор. Если все дело в дуплексе, и он жестко установлен в Вашем коммутаторе, можете изменить настройки **eth0**, выполнив

```
ethtool -s eth0 speed 100 duplex full autoneg off
```

Учтите, что при перезагрузке системы настройки вернутся к первоначальным. Для фиксации настроек укажите опции загрузки модуля для Вашего NIC в **modules.conf**. Если это не решит проблему, есть множество дополнительных вариантов. Ваше сетевое соединение может быть в порядке, а вот нужный Вам сервис может тормозить по множеству причин. Запустите *ifconfig*, и если Вы увидите какие-нибудь Tx/Rx-ошибки – только ли на Вашей машине? Может быть это наблюдается на нескольких компьютерах и дело в коммутаторе? В общем, для решения проблемы нужно сначала отследить и точно определить ее. **ДК**

11 Удаление Linux

В Купил Fedora Core 5 на DVD и проинсталлировал его на свой ноутбук Acer, думая, что буду иметь доступ к установленным программам Windows. И был неправ. Как вы уже поняли, я новичок в Linux. Знаю, что существует *Wine*, позволяющий соединять Windows и Linux. К сожалению, когда я размечал жесткий диск, пропало мое беспроводное соединение с Интернет, и я не могу переподключиться. Также, у меня есть важное ПО на моей Windows XP, к которому больше нет доступа.

Мой вопрос таков: как вернуть таблицу разделов, удалив Fedora Core 5 до тех пор, пока я не буду готов поставить ее? Пытался загрузить *Partition Magic 8.0*, но оно не хочет работать, поскольку имеет формат exe-файла. Вдобавок мой ноутбук поставлялся с предустановленной Windows, и у меня нет дистрибутивного диска. Я посетил множество сайтов, но все решения требуют наличия диска с XP.

Ари [Ari]

О Здесь есть два возможных варианта. Первый – Вы удалили Windows-раздел при установке Fedora Core, выбрав опцию удаления всех разделов. Тогда Вы потеряли Windows, и Вам нужно ее переустанавливать. Вы можете получить инсталляционный CD у производителя или поставщика Вашего ноутбука.

Второй вариант – Windows все еще установлена, но потеряна возможность ее загрузки. Большинство установщиков различных дистрибутивов имеют опцию »

» двойной загрузки с Linux, где Вы можете выбрать, какую ОС запускать, при каждом старте системы. Когда Вы увидите сообщение booting Fedora Core... in n seconds, нажмите на какую-нибудь клавишу; появится меню. Если Windows в нем есть, выберите ее, и Вы снова в Windows. Для удаления загрузчика Fedora Core, чтобы система сразу грузилась в Windows, Вам понадобится спасательный диск Windows. Вам не нужен дистрибутивный компакт-диск, Вы можете все исправить с помощью загрузочных дисков, доступных на www.bootdisk.com. Проще всего скачать образ через какой-нибудь компьютер с Windows и оттуда записать его на дискету.

Загрузитесь с нее и запустите **fixmbr** для восстановления загрузчика Windows и удаления меню *Grub*. Fedora Core останется на диске, но теперь Вы сможете запустить *Partition Magic* и удалить раздел с ней. **НБ**

12 Перемещение через NFS

В Я использую NFS между двумя компьютерами, но слышал, что это не самый быстрый способ транспортировки файлов. Что еще вы можете порекомендовать?

Е. Мэйс [E Mays]

О В самом деле, есть отличный проект *Network Block Device* (http://nbd.sourceforge.net), компилируемый в ядро; по сути, он представляет удаленную файловую систему как локальное устройство. Недостаток в том, что Вы можете смонтировать его только на одной машине. Надеясь, что это для Вас не проблема, я помогу Вам запустить *Network Block Device* – он намного быстрее, чем NFS и гораздо легче в настройке.

Во-первых, *NBD* использует как устройство файл, а не директорию, и Вам нужно создать файл нужного размера. Для создания гигабайтного *NBD*, выполните на сервере:

```
dd if=/dev/zero of=/mnt/nbd-drive bs=1gb count=1
```

Создастся файл **/mnt/remote** размером в 1 ГБ. Далее, сообщите *NBD*-серверу, чтобы он запустился на нужном порту и использовал наш файл. Для примера используем порт 1077:

```
nbd-server 1077 /mnt/ndb-drive
```

Когда он отработает, убедитесь, что модуль *NBD* загружен на клиентской машине, и запустите клиент:

```
modprobe nbd.o
nbd=client 192.168.1.2 1077 /dev/nd0
```

Разумеется, этот IP нужно заменить адресом Вашего сервера. С Вашим *NBD* можно использовать любую файловую систему. Мы же отформатируем его в ext2:

```
mke2fs /dev/nd0
```

Наконец, смонтируем его:

```
mount -text2 /dev/nd0 /mnt/nbd-drive
```

Если на Вашем сервере несколько сетевых карт, Вы можете запустить *NBD* на множестве портов:

```
nbd-server 1077 1078 1079 1080 /mnt/ndb-drive
```

Клиент тоже принимает множество IP-адресов и портов:

```
nbd-client 192.168.1.2 1077 1078 192.168.2.2 1079 1080 /dev/nda
```

ДК

13 Неизвестная web-камера

В Нашел на полке web-камеру с подключением по USB, из маркировки на следует, что это «qb-300». Поиск по Vendor/ProdID (0c45/602a) указывает, что эта камера также известна как Microdia Meade ETX-105EC Camera. Можно ли заставить ее работать в Linux и какие пакеты для этого нужны (я использую Mandriva Linux 2007)? Камера работает под Windows, но только со своими драйверами, которые были утеряны.

svartalf

О Не имея под рукой упомянутой камеры, сложно дать абсолютно точный ответ на Ваш вопрос, но мы все же попробуем. USB-устройство с указанными Вами параметрами поддерживается драйвером *sn9c102*, который входит в последние версии ядра Linux или может быть загружен отдельно по адресу: http://www.linux-projects.org/modules/mydownloads/viewcat.php?cid=2. Для нормальной работы драйвера требуется включить в ядро поддержку подсистемы Video4Linux и USB. Точный список опций ядра, которые необходимо активировать, перечислен в файле **sn9c102.txt**, который можно найти в архиве с исходными текстами ядра в подкаталоге **Documentation/video4linux/** (или же в архиве с самим драйвером, если Вы скачали его отдельно). Прежде чем пересобирать ядро, убедитесь, что в Вашей системе нет готового драйвера (скорее всего, это так): войдите как root и наберите **modprobe sn9c102**. Если Вы получили сообщение об ошибке – значит, модуль отсутствует и должен быть собран. Для этого установите с дисков Mandriva пакет с исходными текстами ядра и инструментарий разработчика – как минимум, *gcc* и *make*.

После того, как драйвер будет установлен и загружен, потребуется установить приложение, которое работает с web-камерой. В принципе, Video4Linux – стандартный интерфейс для видеозахвата и поддерживается многими клиентскими приложениями (см. http://www.exploits.org/v4l/) – хотя бы, тем же *MPlayer*. Если функциональности стандартных приложений для работы с web-камерой типа *GnomeMeeting* или *CamStream* (эти программы включены в Mandriva) для Вас недостаточно, можете загрузить приложение, специально оптимизированное для работы с Вашей камерой, по адресу: www.stolk.org/sonic-snap/. Для Mandriva эту программу можно попытаться собрать из исходных текстов (понадобятся devel-пакеты *FLTK* и *FLAC*) или поискать в сторонних репозиториях.

Следует отметить, что, несмотря на наличие в Linux специализированного драйвера, камера все же может не заработать. Дело в том, что *sn9c102* поддерживает фиксированный набор видеосенсоров, хотя с течением времени этот список расширяется. Если сенсор вашей камеры не поддерживается *sn9c102*, среди сообщений ядра появится строка вроде: usb 4-1: No supported image sensor detected. Единственное, что можно порекомендовать в данном случае (при условии, что Вы используете последнюю версию драйвера) – это связаться с разработчиками и запастись терпением. **ВС**

Вопрос-победитель

Дмитрий Алексеев получает подарочный сертификат на 1000 рублей от интернет-магазина LinuxCenter.Ru! Просим победителя выйти на связь с редакцией: info@linuxformat.ru

Ключ на старт

В Как быстро и оптимально можно клонировать ASP Linux, установленный на 50 компьютеров?

Дмитрий Алексеев

О Эту задачу можно решить несколькими способами. Если компьютеры полностью идентичны (или хотя бы имеют идентичные жесткие диски) и необходимо именно клонировать систему, можно просто посекторно скопировать жесткий диск командой dd или поискать для этого специальные средства. Кроме этого, дистрибутивы семейства Red Hat (в частности, ASP Linux) обладают функцией Kickstart – специальной системой быстрого развертывания однотипной конфигурации на многих компьютерах. Я бы рекомендовал Вам организовать установку по сети и воспользоваться *Kickstart*. Примерный «рецепт» может выглядеть так:

» установите дистрибутив на «эталонный» ПК;

» возьмите с эталонного ПК файл **/root/anaconda-ks.cfg** – сценарий *Kickstart*, созданный на основе параметров текущей инсталляции;

» установите сервер сетевой инсталляции;

» модифицируйте **anaconda-ks.cfg** так, чтобы установка происходила не с CD, а, например, с локального FTP;

» Раздайте **anaconda-ks.cfg**, например, через DHCP. При этом оставшиеся 49 машин удобно загружать через PXE.

Подробное руководство по *Kickstart* на русском языке можно найти по адресу: http://rhd.ru/docs/manuals/enterprise/RHEL-4-Manual/sysadmin-guide/pt-install-info.html **АМ**

Нужна помощь!

› Для точного ответа на ваш вопрос нам нужно знать как можно больше подробностей. Детально опишите конфигурацию системы. Если вы получили сообщение об ошибке, приведите его текст и точно опишите действия, вызвавшие его появление. Если у вас проблемы с оборудованием, то подробно опишите его. Если Linux уже запущен, то выполните в root-терминале следующие команды и прикрепите к письму файл **system.txt**:

```
uname -a >system.txt
lspci >>system.txt
lspci -vv >>system.txt
```

› Пожалуйста, помните, что сотрудники журнала НЕ являются авторами или разработчиками Linux, любых пакетов или дистрибутивов. Зачастую люди, отвечающие за приложения, выкладывают большую часть информации на web-сайты. Попробуйте почитать документацию!

Мы стараемся ответить на все вопросы. Если вы не нашли ответ на свой, это, возможно, потому, что мы уже ответили на похожий вопрос.

Лучшие новинки открытого ПО на планете

LXFHotPicks



В ЭТОТ РАЗ ТОЛЬКО ДЛЯ ВАС: OpenBox » ESpeak » Tellico » PulseAudio » PangZero » Ksirk » Xarchiver » Partition Logic » sshfs » Zile

Оконный менеджер

OpenBox

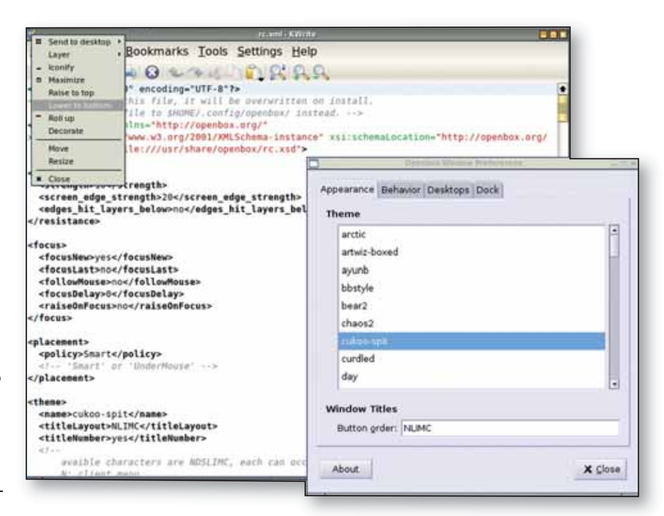
Версия 3.3 **Сайт** www.icculus.org/openbox

Одна из прекрасных отличительных особенностей системы *X Window* – как принято говорить – свобода выбора. Согласно мантре, *X* предоставляет «механизм, а не политику», и не предписывает оформление и поведение окон; это дело оконного менеджера. Но много ли пользователей взаправду выбирают, а не идут на поводу у оконного менеджера, установленного за них в дистрибутиве или окружении рабочего стола?

Если вы ищете новый оконный менеджер (WM), *OpenBox* обязательно следует опробовать. Ранние релизы были на базе *Blackbox*, но в новой версии 3.0 *OpenBox* был переписан с нуля. Как может ожидать пользователь, *OpenBox* поддерживает темы, но больше не может использовать темы *Blackbox* напрямую (инструмент для портирования тем *Blackbox* прилагается).

OpenBox проворен и легок, особенно при сравнении с популярными альтернативами вроде KDE-шного *Kwin* или *Metacity* от Gnome. Благодаря соответствию стандартам, его можно использовать как замену для любого из них (документация объяснят, как). *OpenBox* поддерживает все обычные технологии *WM*, включая уведомления и *Xinerama* (вывод на несколько дисплеев). Верный философии Unix, он не отягчен дополнениями, которые лучше реализовать в виде отдельных инструментов, типа пейджера и стартового меню. Истый минималист может запустить *OpenBox* без раздутого окружения рабочего стола: при этом предоставляется док в духе Window Maker, поддерживающий огромное число апплетов DocApp (см. http://dockapps.org).

Другое, помимо скорости, принципиальное преимущество оконного менеджера вроде *OpenBox* – гибкость настройки. *OpenBox* может полностью перекроить действия мыши и горячие клавиши (и даже обрабатывать «цепочки» клавиш, в стиле *Emacs*), так что вы действительно сможете создать индивидуальный рабочий стол. Другая элегантная особенность *OpenBox* – 'pipe menus'. Главное меню окна не является статичным – его можно генерировать динамически прокачкой результатов исполняемого файла или скрипта. Примеры имеются в изобилии, включая вывод списка сообщений *Gmail* в подменю.



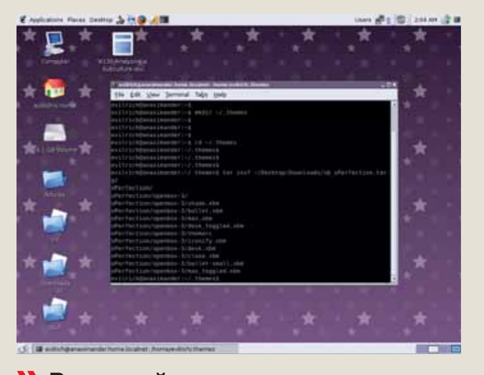
› У *OpenBox* много приятных тем, но кто придумывал им названия? "Cukoo-spit" – «Плевок кукушки»? «Кукушка на вертеле»? «Кукушка выплевывает...» Стоп! Там даже не «кукушка», а «кукша» какая-то!

Для настройки *OpenBox* есть графический интерфейс сторонних производителей, называемый *Obconf* (см. http://tr.openmonkey.com/pages/obconf). Он предоставляет базовый контроль над настройками *OpenBox*: оконной темой, поведением окна при получении фокуса, и т.п. Если вы действительно хотите получить контроль – особенно для правки реакции на действия мыши и нажатия клавиш – придется побороться с файлами конфигурации *OpenBox*.

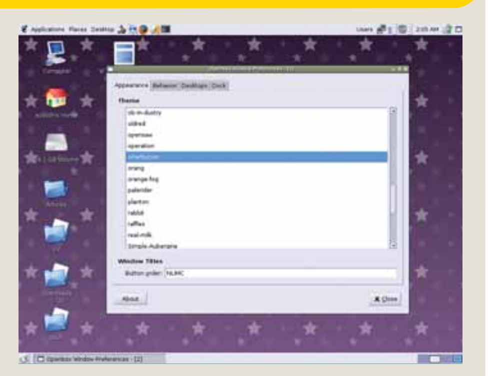
Шаг за шагом: установка тем OpenBox



» Выберите тему
Пройдитесь по различным онлайновым репозиториям тем (попробуйте http://themes.freshmeat.net), выберите тему по вкусу и загрузите ее.



» Распакуйте тему
Пользуясь вашим любимым менеджером архивов или просто командной строкой, распакуйте тему в каталог **.themes** в вашем домашнем каталоге.



» Выберите тему
Запустите *Obconf*, редактор настроек *OpenBox*, перейдите на вкладку Appearance (Вид) и щелкните на имени свежеустановленной темы. Наслаждайтесь.

Синтезатор речи

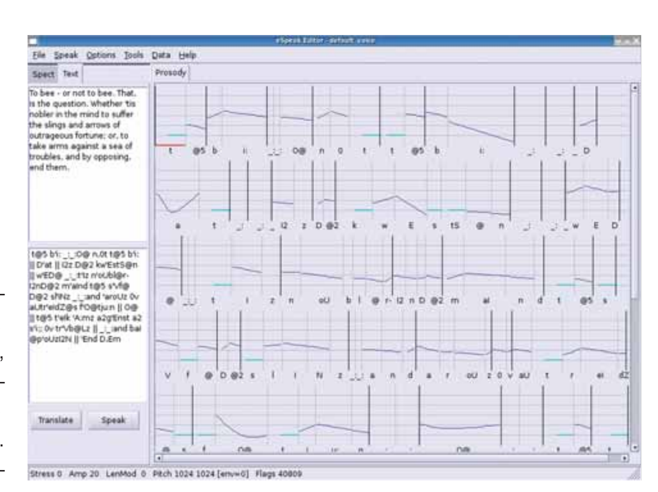
ESpeak

Версия 1.13 **Сайт** http://espeak.sourceforge.net

Синтез речи очаровал некоторых еще во времена первых компьютеров Amiga. Если вам не посчастливилось с ним работать, поясним, что первые версии AmigaOS реализовывали полную поддержку синтеза речи и преобразования «текст-речь» как стандартное устройство AmigaDOS: туда можно было просто скопировать файл, а компьютер его читал. Пусть не как HAL 9000, но вполне прилично, особенно если учесть, что с этим справлялся процессор с частотой 7 МГц и менее 512 КБ ОЗУ. Сейчас, 20 лет спустя, спохватился и весь прочий мир, и синтез речи стал обычным делом, особенно для помощи людям с ослабленным зрением. Так, инструменты чтения с экрана включают и KDE, и Gnome.

Популярен синтезатор с открытым кодом *Festival*, что изначально разрабатывался в Университете Эдинбурга. *ESpeak* – новая GPL-альтернатива. Подход в нем другой, а потому и звук сильно отличается, и требования к памяти и пространству для установки намного меньше. *ESpeak* происходит от синтезатора, написанного для компьютеров Archimedes более 10 лет назад, потому его скромные требования к памяти не удивительны: *Festival*'ю нужна специфичная для голоса библиотека дифонов, занимающая обычно несколько мегабайт, тогда как вся инсталляция *ESpeak* «весит» 950 КБ и поддерживает несколько голосов и языков. Английский, в частности, британский английский, поддерживается лучше всего [поддержка русского находится в зачаточном состоянии: *ESpeak* не делает разницы между твердыми и мягкими согласными, не всегда правильно ставит ударение и т.п., – прим.ред.]. Имеется эмуляция различных региональных британских диалектов, включая акцент центральных графств. Истинное удовольствие слушать, как ваш компьютер лопочет голосом робота-бирмингемца, в отличие от стандартного американского по Хокингу.

> «Истинное удовольствие слушать, как ваш компьютер лопочет голосом робота.»



› ***ESpeak Editor*, опционный графический инструмент для визуализации и управления фонемами, вас позабавит!**

ESpeak можно запускать из командной строки или включать его в ваши приложения путем статической компоновки. Он умеет произносить текст, указанный ему в качестве аргумента, или озвучивать текстовые файлы, и понимает HTML, а также основанный на XML Язык речевой разметки (SSML, Speech Synthesis Markup Language), рекомендованный W3C для приложений синтеза речи. Присутствует опция вывода обрабатываемого текста в виде фонем.

В Gnome нет драйвера для *ESpeak*, и запустить его нельзя. Зато система Text-to-Speech («текст-речь») от KDE не требует написания специальных расширений, и KDE легко настроить на вызов *ESpeak*.

Менеджер коллекций

Tellico

Версия 0.9.10 **Сайт** http://rschultz.ath.cx/code.php

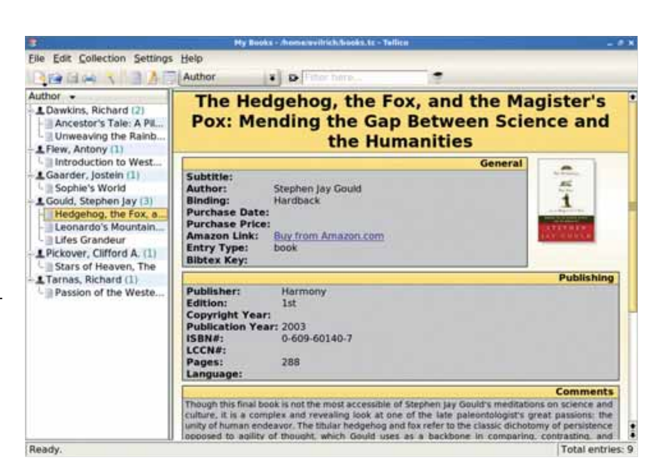
Что подарить хакеру, у которого все есть? Ответ: программу управления его имуществом. Угадали! Название *Tellico* напоминает о провайдере, неумеренно дерущем плату за скверный Интернет, но на самом деле это менеджер коллекций. По умолчанию он знает, как следить за ордами ваших книг, музыки, видео, марок, монет, комиксов, видеоигр… а если у вас нестандартное увлечение, помогут настройки.

Tellico предоставляет полноценный графический интерфейс KDE, и его главное окно позволяет управлять и просматривать отдельные коллекции. Диапазон доступных типов коллекций весьма широк. К примеру, книгу, кроме обычных заглавия, автора, ISBN и так далее, можно снабдить изображением обложки, указать тип переплета, присвоить рейтинг, указать наличие автографа и даже больше.

Прилагаемые инструменты позволяют легко заводить для ваших ценностей атрибуты, в том числе, через встроенный доступ к онлайновым базам данных Например, информацию про книги *Tellico* может скачать с web-сайта Библиотеки Конгресса США, Amazon и http://ISBNdb.com; для музыки – сканировать компакт-диски, а также искать в сети на Yahoo и Amazon. Мы обнаружили, что эти функции не так полезны, как кажутся: возможно, это следствие америкоцентризма или неполноты поддерживаемых баз данных. Что касается книг, например, то мы обнаружили, что *Tellico* смог найти ISBN-записи менее чем для половины содержимого.

Кроме простого слежения за вашим имуществом, *Tellico* позволяет выполнять поиск и печатать отчеты. Еще одна прекрасная функция – генерация цитат из списка книг, также в формате BibTeX, в буфер обмена или напрямую в OpenOffice.org или Lyx. Другая, более приземленная – присмотр за предметами, отданными в аренду. Жаль, что пока не предусмотрена отправка электронных писем должникам, заныкавшим у вас книжоночку. А если серьезно, полезной функцией, которую следовало бы добавить, является интеграция с SANE (Scanner Access Now Easy), чтобы обложки книг, CD, различных записей и тому подобного могли обрабатываться и напрямую импортироваться в базу данных. Ох, ну и наверное, поддержку считывания штрих-кодов для всех этих нудных номеров ISBN.

> «Можно приложить обложку книги, указать рейтинг и прочее.»



› **Способность *Tellico* импортировать данные с Amazon здорово упрощает документирование вашей библиотеки, но обязательна ли ссылка Купить На Amazon?**

LINUX FORMAT Hottest Pick

Звуковой сервер

PulseAudio

Версия 0.9.5 **Сайт** http://pulseaudio.org

Ученые мужи постоянно твердят, что Linux созрел для захвата настольных компьютеров. В общем, они правы. Но продвижению Linux мешают многие второстепенные факторы, включая нормальный способ использования аудиоустройств и, что критично, их совместного использования различными приложениями.

Справедливо будет отметить, что ALSA – Advanced Linux Sound Architecture (Продвинутая Звуковая Архитектура Linux), стандартная среда звуковых драйверов для Linux – сделала несколько полезных шагов в этом направлении с ее расширениями *Dmix* и *Dsnoop*. Они позволяют, соответственно, использовать один аудио-выход или вход одновременно несколькими клиентами ALSA, и хотя ранние версии имели ужасные времена задержки, сейчас эта система вполне работоспособна. Однако эту проблему лучше решать на уровне, не столь близком к оборудованию. ALSA – технология, специфичная для Linux, и завязываться на ее функции означает осложнить портирование. Поэтому, например, и KDE и Gnome используют звуковой сервер, вместо специфичных для платформы звуковых API.

Увы, дорога к нирване рабочего стола завалена трупами неподдерживаемых звуковых серверов. Для Gnome выбран *Enlightened Sound Daemon* или *ESD*, уже несколько лет активно не разрабатываемый, а KDE-проект *Arts* в конце концов не выдержал личностного кризиса и был милосердно усыплен. И что же дальше?

Звуковое решение

PulseAudio, ранее *PolypAudio* – молодой проект, родившийся из останков *ESD* и Arts. Это кросс-платформенный, прозрачно работающий по сети звуковой сервер для POSIX-систем и Windows. *PulseAudio* разработан как автоматическая замена *ESD*, с расширенным набором функций: подсуньте его в Gnome, и приложения спокойно продолжат бибикать и жужжать, ничего не заподозрив.

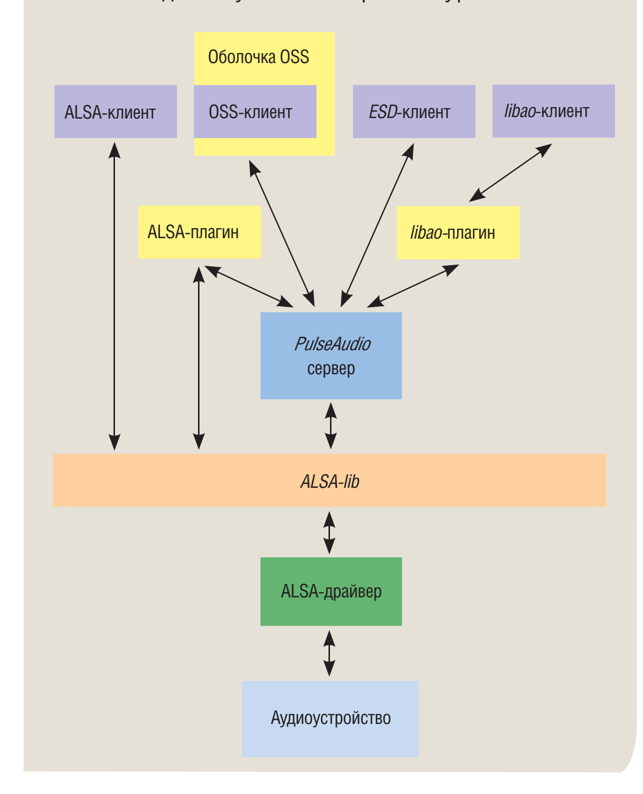
Ключ к гибкости *PulseAudio* – модульная архитектура. Многие из его функций существуют в виде дополнительных модулей, динамически подгружаемых и выгружаемых во время работы. Сервер конфигурируется при помощи простого файла настроек, описывающего модули, намеченные к загрузке, то есть нужные вам функции. После этого, забота о о каждодневной подстройке системы ложится на плечи графических инструментов.

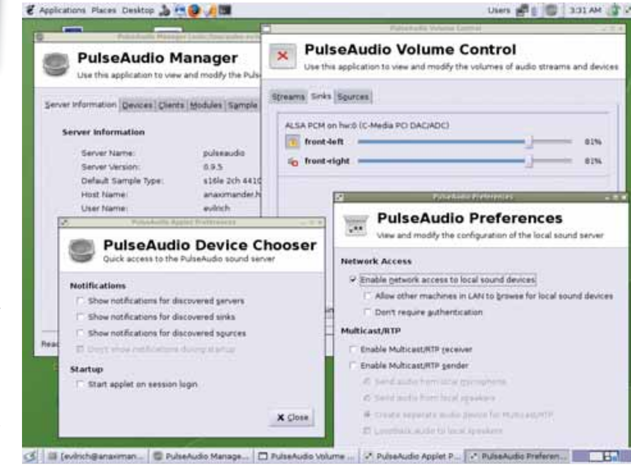
Один из таких необязательных модулей – поддержка HAL для обнаружения локального звукового оборудования. Велите *PulseAudio* использовать этот модуль, и он настроит и использует звуковые карты, поддерживаемые ALSA или OSS, не требуя от вас возни с наладкой.

Другая необязательная функция-модуль *PulseAudio* – поддержка перенаправления звука на удаленные *X*-сессии. Обычно, если вы запускаете *X*-сессию на удаленной машине, звук генерируется и воспроизводится на удаленной машине, а не на устройстве, где отображается *X*-сессия. Что ж, *PulseAudio* имеет решение и на такой случай. Он может прикрепить адрес сервера *PulseAudio* к вашему корневому *X*-окну. Когда приложению потребуется сгенерировать и воспроизвести звуковые данные, он проверит наличие этого свойства у корневого *X*-окна, и если его обнаружит, то возьмет указанный сервер вместо стандартного локального. Это фантастическое решение для *X*-терминалов, при условии, что у вас есть канал достаточной ширины.

Законченное решение

Как взаимодействуют части архитектуры PulseAudio





› Установить *PulseAudio* непросто, зато графических утилит для подстройки его параметров хватает.

Кстати, если вы боитесь задержек при использовании «сетевого» звука, то успокойтесь. *PulseAudio* имеет архитектуру 'zero-copy': если он запущен локально, то ничуть не хуже системы ALSA Dmix. А при сетевой работе, встроенная система измерения задержек *PulseAudio* смягчит любые провалы в сети.

Усваивать и побеждать

Я знаю, о чем вы думаете. Что *PulseAudio* может заменить *ESD*, требует минимума настроек, и вы можете использовать его по сети. Здорово! А как насчет других моих звуковых приложений, не предусматривающих использование *ESD*? Рад, что спросили.

Кроме самой платформы *PulseAudio*, разработчики скрытно работают над различными склейками, которые позволят большинству Linux-приложений общаться с сервером *PulseAudio*. Поэтому *PulseAudio* поставляется с оберткой для OSS, а ALSA 1.0.12 – точнее, *libasound* – обеспечивает поддержку для ввода и вывода *PulseAudio*, и, стало быть, возможность указать сервер *PulseAudio* как устройство ALSA по умолчанию (вы, вероятно, знаете, что ALSA сама по себе включает совместимость с OSS).

Помимо ALSA- и OSS-приложений, разработчики позаботились об интерфейсах ко многим аудио-приложениям и библиотекам. В *GStreamer*, *libao* и *Xine* доступны интерфейсы (back-ends) для прямого общения с *Pulse*. Со временем их будет еще больше, включая драйвер SDL. Есть даже порт для *MPlayer*.

Если вам надо адаптировать свое собственное приложение, то к *PulseAudio* прицепиться легко. Тут есть два API: простой синхронный API и более сложный асинхронный. Работа с ними в любом случае намного проще написания приложения для ALSA.

На текущий момент, минус *PulseAudio* – нудная и длительная процедура сборки и установки (в основном из-за множества библиотек зависимостей). Когда главные дистрибутивы начнут включать его – а они, вероятно, начнут, если верны слухи и *Pulse* будет звуковым сервером в Gnome 2.18 – то и это препятствие канет в небытие.

HotGames Развлекательные приложения

Аркада

PangZero

Версия 0.14 **Сайт** http://apocalypse.rulez.org/pangzero

Что нужно для создания классической аркады? Обманчиво простой ход игры и, естественно, крутая графика. Ну и еще сценарий, слабо связанный с реальностью. Если рецепт таков, то *PangZero*, клон *SuperPang*, несомненно заслуживает признания.

Ваш герой в *PangZero* выглядит как отброс из South Park, вооружен гарпуном и обязан путешествовать по миру скачущих мячей, низвергающихся с неба. Зачем? Кто знает или дерзает знать! Неизвестно почему, прикосновение мячей смертельно, так что пытайтесь избегать их, насаживайте эти чертовы штуки на дротики. Подстрелите большой мяч – он развалится на два; загарпуньте половинку – и она вновь разделится. Только когда мячи-потомки достаточно измельчают, попадание заставит их исчезнуть.

Идея для игры, может, и тупая, но жутко втягивающая и безумно сложная. Для добавления интереса к этому дурацкому занятию имеется несколько видов мячей: голубой водяной мяч делится при каждом отскоке, удар сейсмического мяча приводит к землетрясению, колдовской мяч двигается по хитрой траектории, пренебрегая законами гравитации, и т.д. Если вы ухлопаете мерцающий мяч или зеленый супер-мяч, то получите передышку – мячи остановятся, а время пойдет обратно, предоставив вам прекрасный момент для очистки экрана. Иногда появляется подмога: например, можно подхватить пулемет, более скорострельное оружие, чем гарпун.

> «Играя с другом, а то и с пятью, выжить намного проще.»



› **Не стреляйте в черный шар смерти! Убить его нельзя: он только размножится.**

Как и в классической Asteroids, штука в том, чтобы не засорить экран мячами-потомками, а это обязательно произойдет при беспорядочной пальбе. Старайтесь также заработать как можно больше свободного времени. Если вам не повезет, пригласите друга, а то и пятерых (да-с! игра поддерживает до шести игроков одновременно) – выжить будет намного проще.

Стратегия

Ksirk

LINUX FORMAT Hottest Pick

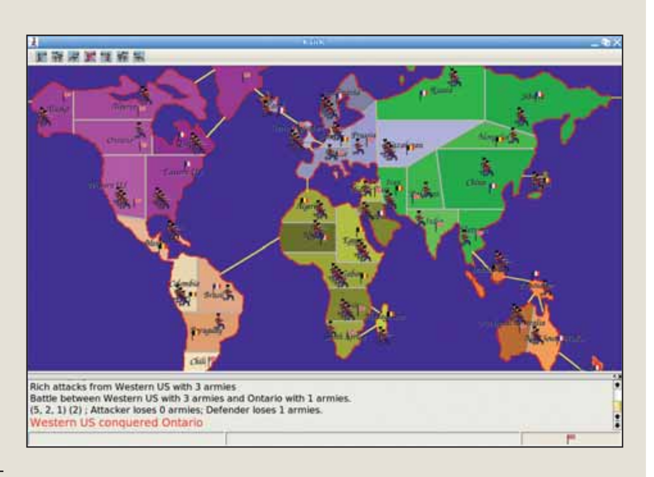
Версия 1.6 **Сайт** http://home.gna.org/ksirk

Каждый желает править миром – в чем пытался убедить нас рок-хит восьмидесятых Tears For Fears от наших братьев Bathonians. Если вы действительно страдаете манией величия, можете чуток попрактиковаться на настольной игре *Risk*. Эта классическая смесь стратегии и удачи за многие годы видела много компьютерных версий, официальных и не очень, и даст выход вашему комплексу Наполеона, даже когда вы ликвидируете всех подручных соперников.

В Linux появилось несколько бесплатных неофициальных игр, вдохновленных Risk, и *Ksirk* – одна из них. Не будучи наипрекраснейшей из игр-завоеваний с открытым кодом, *Ksirk* имеет весьма близкий к оригиналу ход игры, компьютерных противников и поддержку игры по сети. Победа над компьютерным или неизвестным удаленным игроком, бесспорно, не сравнится со зрелищем запуганного врага, загнанного в угол вашим неумолимым наступлением, но оно ненадолго утолит вашу жажду мирового господства.

Ksirk предоставляет простой, основанный на KDE (а вы что подумали, видя букву 'K'?) графический интерфейс; его основной компонент, естественно, карта мира – ваше поле битвы. В качестве примера поставляется Воображаемый Мир (BubbleWorld). Подобно своему прототипу, *Ksirk* позволяет играть в игры-миссии или простой захват мира. Локальная игра-миссия, очевидно, требует, чтобы ваши оппоненты отвернулись, когда ваша миссия отобразится на экране.

Интерфейс пользователя слегка неуклюж. Например, чтобы открыть сражение, вы должны сначала выбрать количество атакующих армий, затем нажать и придержать кнопку мыши на стране-агрессоре, а затем перенести курсор на страну – цель нападения (было бы удобнее сначала выбрать атакующую страну). Но не беда! Визуальные украшения вроде анимированных пушек, стреляющих друг в друга, пока вы атакуете, помогают полноте восприятия виртуального побоища, и вы ощутите в ноздрях сладкий запах победы.



› **Мечта Эффи сбылась: Мексика захватила весь континентальный массив Центральной Америки…**



Менеджер архивов

Xarchiver

Версия 0.4 **Сайт** http://xarchiver.xfce.org

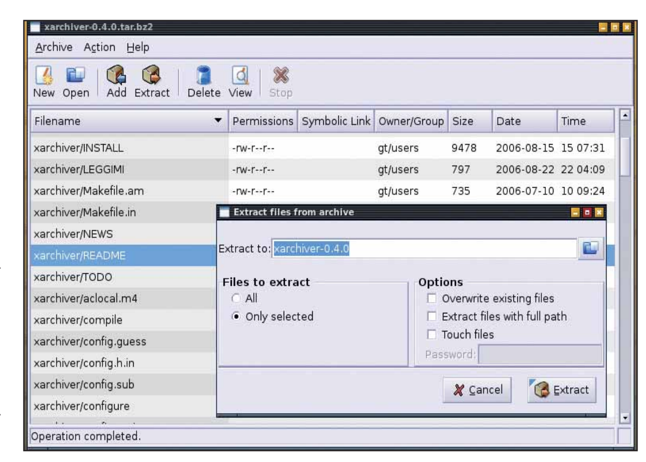
Джузеппе Торелли (Giuseppe Torelli), разработчик *Xarchiver* – *GTK*-оболочки для управления различными типами файловых архивов – славно потрудился со времени последнего рассмотрения его программы в LXF75: не прошло и года, а *Xarchiver* намного похорошел.

Для начала, напомним кратко: *Xarchiver* предоставляет модель, схожую с управлением файлами для создания, просмотра и извлечения, а также добавления файлов в различные популярные файловые архивы. Вы, конечно, можете использовать для этого командную строку, но новичками графический интерфейс осваивается быстрее. Эта программа не так объемна, как альтернативы от KDE и Gnome, и будет полезна для легковесных рабочих столов.

Версия 0.3 умела управляться с вездесущими tar-архивами (сжатыми и при помощи *gzip*, и *bzip2*), а также архивами zip, arj, rar, 7zip и RPM. Новый релиз добавляет еще и поддержку ISO-образов (правда, только для чтения). Напомним, что по своей сути, *Xarchiver* – визуальная надстройка над набором архиваторов командной строки, и манипулировать архивами некоторого типа можно, только если у вас установлен архиватор данного типа, к которому и произойдет обращение. То есть, например, для работы с *zip*-архивами необходимо наличие архиватора zip.

В версии 0.4 наконец-то переделан интерфейс пользователя: GUI стал более продуман и прост, и намного ускорился. Вы теперь можете перетаскивать файлы в и из главного окна *Xarchiver*, а для всех команд меню *Xarchiver* определены горячие клавиши. Добавлена возможность тестирования целостности архива, а также удобный диалог свойств архива, показывающий число файлов и каталогов в архиве, их размер, дату создания, степень сжатия и так далее. Представление содержимого архива в виде списка теперь может отображать символьные ссылки, если они поддерживаются архиватором.

> «Xarchiver наконец-то переделал интерфейс пользователя.»



› *Xarchiver*-овские диалоги Add (добавить) и Extract (Распаковать) файлы обычно поддерживают кучу опций.

Вы все еще можете счесть, что *Xarchiver* не вполне отвечает вашим нуждам. Мы бы предпочли видеть дерево вместо простого списка содержимого архива, и было бы здорово, если бы *Xarchiver* понимал LHA. Другие могут решить, что интегрированная поддержка архивных файлов в KDE и Gnome работает более гладко. Однако умное распознавание типа архива в *Xarchiver* (не просто по расширению файла) и поддержка более экзотических особенностей, предоставляемых некоторыми типами архиваторов, завоюют ему сторонников.

Редактор разделов

Partition Logic

Версия 0.63 **Сайт** http://partitionlogic.org.uk/index.html

Переразбивку своих дисков ненавидят все. Не только потому, что это чревато бедой – не нарушу ли я имеющиеся разделы, смогу ли восстановиться с резервной копии? – но и потому, что в сфере Open Source нет хорошей альтернативы коммерческому *PartitionMagic*. Да, есть *Parted*, но многие пользователи скорее сядут в зубоврачебное кресло, нежели воспользуются командным редактором разделов.

Но не теряйте надежды. *Partition Logic* – попытка создать клон *PartitionMagic* с открытым кодом. Он поставляется в виде ISO-образа (имеется также издание на дискете), который вы записываете на CD-R, а затем используете, загружая с него свой компьютер, чтобы попасть в простой, но функциональный, основанный на GUI редактор разделов.

Любопытна архитектура *Partition Logic*. Вместо создания программы для Linux, запускаемой в легком Linux-дистрибутиве, автор, Эндрю Мак-Лафлин [Andrew McLaughlin], взял за основу свою операционную систему Visopsys – легкую, быструю многозадачную ОС с интегрированным оконным инструментарием. Редактор разделов *Partition Logic*, вообще-то – менеджер дисков из Visopsys, и это дает ему ряд преимуществ. Он занимает невероятно мало места на диске – ISO-образ менее 8 Мб – и, соответственно, скромен в системных требованиях. Он запустится на любом компьютере с процессором Pentium, 32 МБ ОЗУ и VESA2-совместимой видеокартой. С новыми устройствами, однако, все не так удачно. Плохо поддерживается Serial ATA (*Partition Logic* не смог найти ни одного диска на тестовой машине с чипсетом NForce), а драйверов USB Visopsys вообще не имеет, так что ввод должен осуществляться через PS2-периферию.

Премии за дизайн графическому инструментарию Visopsys не дадут, и редактор разделов тоже минималистичен. Например, размер раздела нужно вводить в текстовый виджет, без каких-либо графических средств типа ползунка. *Partition Logic* понимает только стандартные MBR-таблицы разделов DOS и ничего более экзотичного вроде таблиц разделов *BSD, Sun или EFI. Поддержка файловых систем также ограничена: не поддерживаются логические тома Linux, можно менять размер только NTFS-разделов и нельзя форматировать ReiserFS или Ext3.

Но если помнить все эти ограничения, *Partition Logic* – полезный пакет, особенно для запуска на старом оборудовании. И кто знает – может, стоит выказать Мак-Лафлину свою признательность, и он расширит функциональность?

> «Запускается практически на любом компьютере с Pentium и 32 МБ ОЗУ.»

› Более крупный и четкий моноширинный шрифт облегчил бы работу с интерфейсом *Partition Logic*.

Текстовый редактор

Zile

Версия 2.2.19 **Web** http://zile.sourceforge.net

В обществе не полагается говорить о трех вещах: религии, политике и вашем любимом текстовом редакторе. Но какой бы стороны вы ни придерживались в противостоянии *Emacs* против *Vi*, вы, вероятно, согласитесь, что *Emacs* все-таки великоват для быстрой работы. Его загрузка может занять больше времени, чем выполнение работы вручную!

А не надо ли на всякий случай иметь редактор, по виду, по вкусу и в работе похожий *Emacs*, но размером, скажем, до 100 КБ? Что ж, *Zile* (*Zile Is a Lossy Emacs*, «*Zile* – это отощавший *Emacs*») – именно такой зверь. *Zile* полностью подражает интерфейсу *Emacs*, от горячих клавиш команд до поддержки мульти-буфера для вырезания и вставки. Фактически, руководство к *Zile* отличается от *Emacs* разве что поиском и заменой имени. *Zile* даже поддерживает режим автозаполнения в духе *Emacs* (word wrap), который хорош для настукивания простых текстовых документов.

Zile не поддерживает Unicode и не имеет подсветки синтаксиса, и вам вряд ли захочется использовать его для серьезной работы над исходным кодом – но он для этого и не предназначен. Если вам нужны подобные продвинутые функции, используйте что-то посолиднее, типа *Jed*. А не нужны – попробуйте *Zile*.

› **Мини-буфер в стиле *Emacs* с завершением команд, причем в крошечном исполняемом файле. *Zile* слишком хорош, чтобы быть правдой.**

Файловая система

sshfs

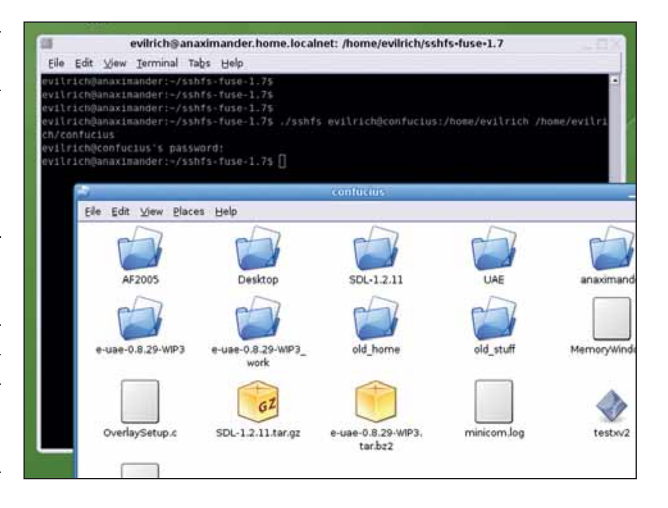
Версия 0.82 **/** 0.48b **Сайт** http://fuse.sourceforge.net/sshfs.html

Традиционные файловые системы Linux включены в само ядро. В принципе, это хорошая идея, особенно для производительности. Однако, не втягиваясь в утомительные споры на тему «монолит против микроядра», бывают ситуации, когда файловые системы пространства пользователя также имеют значение, особенно для более эзотерических типов файловых систем. Поэтому был создан проект Fuse – драйвер ядра и соответствующая инфраструктура, позволяющая записывать файловые системы пространства пользователя под Linux.

Прекрасным примером правильного применения платформы Fuse является *sshfs*: вы можете монтировать каталоги с любого удаленного SSH-сервера, используя SSH File Transfer Protocol. Клиент *sshfs* не изобретает велосипед, а использует стандартный ssh – инструмент командной строки. Ну и как бы вы это сделали из ядра?

Имеется несколько минусов *sshfs*, о которых стоит сказать. Отдельная точка монтирования *sshfs* годится только для одного пользователя. Если вы хотите смонтировать файлы, разделяемые несколькими пользователями, то лучше перейти на NFS. Но совместное использование файлов посредством *sshfs* имеет несколько очевидных преимуществ. Во-первых, не нужна сложная настройка *Samba* или NFS-сервера; основные системы уже имеют настроенный SSH-сервер. Во-вторых, монтирование совместно используемых ресурсов на базе *sshfs* не требует привилегий root – необходимо лишь разрешение использовать устройство **/dev/fuse** (достигаемое простым добавлением себя в группу 'fuse'). Гениально. LXF

› **Молить сисадмина о разрешении не придется: sshfs обходится без root-доступа.**



Также выпущены

Новые и обновленные приложения, также заслуживающие внимания...

» **Amanda 2.5.1** Популярная система сетевого резервирования и архивирования http://sourceforge.net/projects/amanda

» **Amarok 1.4.3** Многогранный аудиопроигрыватель для KDE http://amarok.kde.org

» **Awffull 3.6.1** Улучшенный анализатор журналов web-сервера, ответвление Webalizer www.stedee.id.au/awffull

» **Eterm 0.9.4** Богатый функциями эмулятор X-терминала www.eterm.org

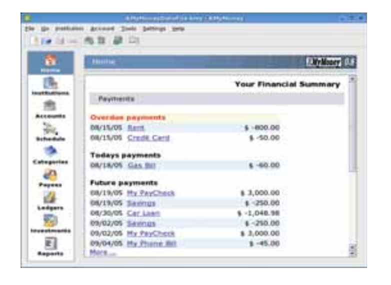
» **Gallery 2.12** Удобный фотоальбом на базе web http://gallery.sourceforge.net

» **GNUstep 1.13.0** Клон платформы NextSTEP/Cocoa с открытым кодом www.gnustep.org

» **GraphThing 1.3** Визуальный инструмент теории графов http://graph.seul.org

» **GTKLife 5.0** Изучите Жизнь Конуэя при помощи мощного GTK-интерфейса http://ironphoenix.org/tril/gtklife

» **K3b 0.12.17** Легкий в использовании пакет для записи CD/DVD www.k3b.org



› ***KMyMoney* ответит на вопрос, куда деваются деньги...**

» **KMyMoney 0.8.5** Персональный финансовый менеджер для KDE http://kmymoney2.sourceforge.net

» **LifeSaver 0.3** Защита редактора вашего браузера от сбоев http://freshmeat.net/redir/livesaver/65439/url_homepage/livesaver.24inch.org

» **OpenPHPNuke 2.3.7** Ответвление оригинальной PHP-системы управления контентом www.openphpnuke.com

» **Rafkill/Raptor 1.2.2** Стрелялка с вертикальной прокруткой в духе старой школы http://raptorv2.sourceforge.net

» **UNFS3 0.9.15** Сервер пользовательского пространства NFS v3 http://unfs3.sourceforge.net

» **Xvkbd 2.8** Виртуальная клавиатура для X-дисплеев http://homepage3.nifty.com/tsato/xvkbd

› ***Xvkbd*: для ручных устройств и при невозможности работы с клавиатурой.**

LXF DVD86

Диск с двойной загрузкой, полный дистрибутивов для новичков и ассов!



Майк Сондерс любовно подбирает содержимое диска Linux Format, а также поддерживает сайт **www.linuxformat.co.uk**.

Хотели этого? Вот вам!

Две новости этого месяца: во-первых, спасибо всем, кто откликнулся на наш онлайн-опрос после редизайна LXF84. Мы очень ценим обратную связь с вами, она помогает нам создавать для вас самые лучшие диски!

Вы сообщили, что больше всего вам хотелось бы видеть на диске полные дистрибутивы, журнал в формате PDF и альтернативные операционные системы. Мы, естественно, удовлетворяем ваши желания, и на диске этого месяца – два превосходных новых релиза дистрибутивов: Gentoo 2006.1 для опытных пользователей и PCLinuxOS для домашних компьютеров и новичков в Linux. А еще – подборка статей журнала в формате PDF, дистрибутивы поменьше и альтернативные операционные системы с открытым исходным кодом.

А теперь вторая новость: после 45 выпусков я передаю эстафету HotPicks способнейшему Ричарду Драммонду [Richard Drummond], автору статей в *Linux Format*. Это были славные годы: пробовать кучи новых программ, прочесывать Интернет в поисках скрытых сокровищ и периодически копаться в коде, чтобы заставить какую-нибудь программу работать! А главное – восхищенно наблюдать, как крепнет и распространяется талант и благосостояние открытого кода, с его армией разработчиков, вынашивающих собственные проекты. Но настала пора прерваться, и теперь с лучшими новыми приложениями в разделе HotPicks вас будет знакомить Рич.

Если у вас есть предложения или комментарии – пожалуйста, черкните мне пару строк. mike.saunders@futurenet.co.uk

Содержание DVD

ЖУРНАЛ
- **3D Games** .... Файлы из учебника Ogre.
- **Gtk** .... Файлы из учебника Gtk
- **Список статей** .... Список статей предыдущих выпусков **LXF**.
- **Linus** .... Линус произносит "Linux".
- **PDFs** .... Статьи **LXF**.
- **Unix API** .... Файлы из учебника Unix API
- **Сравнение** .... Файловые менеджеры.

РАБОЧИЙ СТОЛ
- **Buddi** .... Финансовый менеджер.
- **DWM** .... Dynamic Window Manager.
- **Extcalc** .... Научный калькулятор.
- **Faces** .... Инструмент управления проектами.
- **Gnome** .... Исходный код рабочего окружения
- **Xpdf** .... Просмотрщик PDF.

РАЗРАБОТКА
- **Glade** .... Программа для создания GUI .
- **Kommander** .... Среда разработки.
- **Package Wizard** .... Программа для создания пакетов.
- **Subversion** .... Система контроля версий.

ДИСТРИБУТИВЫ
- **DeLi Linux** .... Легковесный дистрибутив.
- **Gentoo** .... Дистрибутив с акцентом на производительности.
- **Haiku** .... Клон BeOS.
- **Knoppix** .... Живой дистрибутив.
- **PCLinuxOS** .... Настольный дистрибутив.
- **ReactOS** .... Windows-подобная ОС.
- **SLS** .... Древний дистрибутив.
- **Ubuntu** .... Настольный дистрибутив.

ИГРЫ
- **Freedoom** .... Doom с открытым кодом.
- **Max Fighter** .... Космическая стрелялка.
- **Nikwi** .... Консольная игра.
- **Toppler** .... Консольная игра.
- **X-Moto** .... Гонки.

ГРАФИКА
- **Gwenview** .... Программа для просмотра изображений под KDE
- **Inkscape** .... Редактор векторной графики.
- **Xara Xtreme** .... Графический редактор.

СПРАВКА
- **Rute** .... Руководство по администрированию Linux.

HOTPICKS
- **ESpeak** .... Синтезатор речи.
- **Ksirk** .... Игра в жанре стратегии.
- **OpenBox** .... Менеджер окон.
- **PangZero** .... Игра в жанре «экшн».
- **Partition Logic** .... Менеджер разбиения диска.
- **PulseAudio** .... Аудио сервер.
- **sshfs** .... Файловая система на основе SSH.
- **Tellico** .... Менеджер коллекций.
- **Xarchiver** .... Менеджер архивов.
- **Zile** .... Текстовый редактор.

ИНТЕРНЕТ
- **CheckGmail** .... Программа уведомления о получении электронных сообщений.
- **Firefox** .... Web-браузер.
- **Galeon** .... Web-браузер на основе Gecko.
- **Linphone** .... Internet телефон.

БЕЗОПАСНОСТЬ
- **Aegis** .... Сканнер вирусов.
- **Nmap** .... Сканнер портов.

СЕРВЕР
- **Cherokee** .... Web-сервер.
- **Contineo** .... Система управления документами.

ЗВУК
- **Audacious** .... Музыкальный плейер.
- **Banshee** .... Музыкальный плейер Gnome.
- **Herrie** .... Музыкальный плейер командной строки.
- **KRadio** .... GUI-радио под KDE.

СИСТЕМА
- **KWifi Selector** .... Программа выбора сети.
- **Qemu** .... Эмулятор CPU и ПК.
- **Wine** .... Слой совместимости с Windows.

ГЛАВНОЕ
- **Avifile** .... Библиотека чтения/записи AVI файлов.
- **Bash** .... Командная оболочка.
- **CheckInstall** .... Программа для создания двоичных пакетов.
- **Coreutils** .... Утилиты командной строки.
- **CSV** .... Список файлов, содержащихся на диске.
- **glib** .... Низкоуровневая библиотека ядра.
- **glibc** .... Библиотека GNU C.
- **GTK** .... Инструментарий GUI.
- **Jigdo** .... Программа для создания ISO-образов.
- **Kernel** .... Свежий релиз ядра Linux.
- **libsigc** .... Система обратных вызовов для C++.
- **libXML** .... Анализатор и набор инструментов XML.
- **Ncurses** .... Оконный инструментарий текстового режима.
- **Python** .... Язык программирования.
- **Rawrite** .... Запись изображений на дискеты.
- **SBM** .... The Smart Boot Manager (менеджер загрузки).
- **SDL** .... Библиотека мультимедиа.

Информация о диске

Внимательно прочтите это перед тем, как использовать DVD-диск.

ЧТО-ТО ПОТЕРЯЛИ?

Часто случается, что новые программы зависят от других программных продуктов, которые могут не входить в текущую версию вашего дистрибутива Linux.

Мы стараемся предоставить вам как можно больше важных вспомогательных файлов. В большинстве случаев, последние версии библиотек и другие пакеты мы включаем в каталог «Essentials» (Главное) на прилагаемом диске. Поэтому, если в вашей системе возникли проблемы с зависимостями, следует заглянуть именно туда.

ФОРМАТЫ ПАКЕТОВ

Мы стараемся включать как можно больше различных типов установочных пакетов: RPM, Deb или любые другие. Просим вас принять во внимание, что мы ограничены свободным пространством и доступными бинарными выпусками программ. По возможности, мы будем включать исходные тексты для любого пакета, чтобы вы смогли собрать его самостоятельно.

ДОКУМЕНТАЦИЯ

На диске вы сможете найти всю необходимую информацию о том, как устанавливать и использовать некоторые программы. Пожалуйста, не забывайте, что большинство программ поставляются вместе со своей документацией, поэтому дополнительные материалы и файлы находятся в соответствующих директориях.

ЧТО ЭТО ЗА ФАЙЛЫ?

Если вы новичок в Linux, вас может смутить изобилие различных файлов и расширений. Так как мы стараемся собрать как можно больше вариантов пакетов для обеспечения совместимости, в одном каталоге часто находятся два или три файла для различных версий Linux, различных архитектур, исходные тексты и откомпилированные пакеты. Чтобы определить, какой именно файл вам нужен, необходимо обратить внимание на его имя или расширение:

имя_программы-1.0.1.i386.rpm – вероятно, это бинарный пакет RPM, предназначенный для работы на системах x86;

имя_программы-1.0.1.i386.deb – такой же пакет, но уже для Debian;

имя_программы-1.0.1.tar.gz – обычно это исходный код;

имя_программы-1.0.1.tgz – тот же файл, что и выше по списку: «tgz» - это сокращение от «tar.gz»;

имя_программы-1.0.1.tar.bz2 – тот же файл, но сжатый bzip2 вместо обычного gzip;

имя_программы-1.0.1.src.rpm – также исходный код, но поставляемый как RPM-пакет для упрощения процесса установки;

имя_программы-1.0.1.i386.FC4.RPM – бинарный пакет RPM для x86, предназначенный специально для операционной системы Fedora Core 4;

имя_программы-1.0.1.ppc.Suse9.rpm – бинарный пакет RPM, предназначенный специально для операционной системы SUSE 9.x PPC;

имя_программы-devel-1.0.1.i386.rpm – версия для разработчиков.

Если диск не читается...

Это маловероятно, но если все же прилагаемый к журналу диск поврежден, пожалуйста, свяжитесь с нашей службой поддержки по электронной почте: disks@linuxformat.ru



LXFDVD86

Декабрь 2006

LINUX FORMAT 8 GIGABYTE DVD





PDF мания

большой архив статей в формате PDF из предыдущих выпусков журнала

- Серия "Что за штука..."
- Серия учебников по Metapost
- Серия учебников по PAW
- Серия учебников по Python

А также

- Ubuntu 6.10: новая версия популярного дистрибутива
- PCLinuxOS 0.93: форк Mandriva Linux
- DeLi Linux: дистрибутив для старых машин
- SLS: посмотрите, каким был Linux в 1993 году!
- Max Fighter: красивый космический шутер

LiveCD Испытайте последние версии Linux прямо с этого диска!



Gentoo Linux 2006.1

Экстримальная производительность и конфигурируемость - выбор продвинутых пользователей.





Содержание LINUX FORMAT DVD Декабрь 2006

Сторона 1

Рабочий стол
Buddi – программа для учета финансов
dwm – легкий оконный менеджер
Extcalc – продвинутый инженерный калькулятор
Faces – менеджер проектов
Gnome – рабочий стол для Unix
Xpdf – утилита для чтения PDF-файлов

Разработка
Glade – программа для построения графических интерфейсов
Kommander – RAD для bash
Package_Wizard – утилита для быстрого создания пакетов
Subversion – система контроля версий, аналог CVS

Дистрибутивы
DeLi Linux – дистрибутив для устаревших компьютеров
Gentoo – популярнейший source-based дистрибутив Linux
Haiku – открытый клон BeOS
Knoppix – многофункциональный LiveCD на базе Debian
PCLinuxOS – настольный дистрибутив на базе Mandriva
ReactOS – открытый клон Windows NT
SLS – один из первых дистрибутивов Linux
Ubuntu – новая версия популярной системы

Игры
FreeDoom – полностью свободная реализация Doom
Max_Fighter – космический шутер
Nikwi – аркадная игра для детей
Toppler – открытый клон Tower Toppler
X-Moto – двумерный мотосимулятор с необычной физикой

Графика
GwenView – просмотрщик изображений для KDE
Inkscape – профессиональный векторный редактор
Xara Xtreme – открытый порт векторного редактора от Xara для Linux

Справка
RUTE – книга по системному администрированию Linux

Горячие новинки
eSpeak - синтезатор голоса
KsirK - компьютерная версия стратегической игры Risk
Openbox - легкий оконный менеджер
Pang Zero - улучшенный клон аркадной игры Super Pang
Partition_Logic - утилита для управления разделами жесткого диска
PulseAudio - звуковой сервер для POSIX-систем и Win32.
SSHFS - клиент для монтирования SSH-ресурсов
Tellico - программа для организации коллекций
Xarchiver - графический интерфейс к архиваторам
Zile - маленький клон редактора Emacs

Интернет
CheckGmail – апплет для проверки почты на Gmail
Firefox – новая версия популярного браузера
Galeon – легкий браузер на базе Gecko
LinPhone – VoIP-клиент

Безопасность
Aegis – антивирусный сканер
Nmap – утилита для аудита безопасности сети

Сервер
Cherokee – веб-сервер
Contineo – система управления документами

Звук
Audacious – форк beep-media-player
Banshee – продвинутый аудиопроигрыватель на Mono
Herrie – плеер для командной строки
Kradio – радиоприемник для KDE

Система
KwifiSelector – утилита для управления беспроводными соединениями
Qemu – открытый эмулятор ПК
Wine – свободная реализация WinAPI для Unix-систем

Сторона 2

Slackware 11.0 – старейший из ныне существующих дистрибутивов Linux
Slax 5.1.8.1 – LiveCD на основе Slackware Linux

Комментарии? Присылайте ваши мысли и предложения по электронной почте: info@linuxformat.ru
Пожалуйста, ознакомьтесь с опубликованной в журнале инструкцией перед использованием данного диска.
Настоящий диск тщательно тестировался и проверялся на всех стадиях производства, однако, как и в случае с любым новым ПО, мы рекомендуем вам использовать антивирусный сканер. Мы также рекомендуем всегда иметь под рукой актуальную резервную копию данных вашего жесткого диска. К сожалению, редакция Linux Format не может принимать на себя ответственность за любые повреждения, разрушения или иные убытки, которые может повлечь за собой использование этого DVD, представленных на нем программ или данных. Перед тем, как устанавливать какое-либо ПО на компьютер, подключенный к сети, проконсультируйтесь с сетевым администратором.

Дефектные диски. В маловероятном случае обнаружения дефектов на данном диске, пожалуйста, обращайтесь по адресу: disks@linuxformat.ru

Тираж изготовлен на Уральском электронном заводе., 620066, Россия, г. Екатеринбург, ул. Комвузовская 17-203, Лицензия МПТР России ВАФ № 77-15

Поставляется вместе с журналом LINUXFORMAT номер 12(85) Декабрь 2006





Создание установочных дисков при помощи cdrecord

Самый быстрый способ записать ISO-образ на чистую матрицу – это *cdrecord*. Для всех перечисленных ниже действий потребуются права root. Для начала определите путь к вашему устройству для записи дисков. Наберите следующую команду:

```
cdrecord -scanbus
```

После этого на экране терминала должен отобразиться список устройств, подключенных к вашей системе. SCSI-адрес каждого устройства представляет собой три числа в левой колонке, например, 0,3,0. Теперь вы можете с легкостью записать образ на диск:

```
cdrecord dev=0,3,0 -v /путь к образу/image.iso
```

Чтобы упростить дальнейшее использование *cdrecord*, сохраните некоторые настройки в файле **/etc/default/cdrecord**. Добавьте по одной строке для каждого устройства записи (вероятно, в вашей системе присутствует всего одно такое устройство):

```
Plextor= 0,3,0 12 16M
```

Первое слово в этой строке – это метка, затем, после адреса SCSI-устройства вы должны указать скорость и размер буфера. Теперь вы можете заменить SCSI-адрес в командной строке на выбранную вами метку. Все будет еще проще, если вы добавите следующее:

```
CDR_DEVICE=Plextor
```

Все, что вам теперь нужно для записи ISO-образа – это набрать команду

```
cdrecord -v /path/to/image.iso
```

Если вы не из числа любителей командной строки, в таком случае вам придет на помощь утилита *gcombust*. Запустите ее из-под root, выберите вкладку Burn и ISO 9660 Image в верхней части окна. Введите путь к образу, который вы хотите записать на диск, и смело нажимайте на Combust!. Пока ваш образ пишется на диск, можете выпить чашечку кофе.

Другая ОС?

Вам не обязательно использовать Linux для записи компакт-диска. Все необходимые файлы уже включены в ISO-образ. Программы вроде *cdrecord* просто переносят данные на чистую матрицу. Если у вас нет устройства для записи дисков, можно найти того, у кого оно есть, и записать диск на его компьютере. На нем может стоять Windows, Mac OS X, AmigaOS, или любая другая ОС.

Нет устройства для записи дисков?

А что, если у вас нет устройства, с помощью которого можно было записать образ на диск? Вы знаете кого-либо с таким устройством? Вам не обязательно использовать Linux для записи дисков, подойдет любая операционная система, способная распознать пишущий привод (см. выше).

Некоторые дистрибутивы умеют монтировать образы дисков и выполнять сетевую установку или даже установку с раздела жесткого диска. Конкретные методы, конечно, зависят от дистрибутива. За дополнительной информацией обращайтесь на web-сайт его разработчика. **LXF**

Дистрибутив Linux

Gentoo 2006.1

Когда в марте 2002 появился Gentoo Linux 1.0, немногие пользователи Linux сумели оценить по достоинству дистрибутив, предлагающий собирать все своими руками прямо из исходных текстов. Однако вскоре стало ясно, что степень настраиваемости, доступная в таком дистрибутиве, нужна не только хакерам – она действительно полезна. Не нужна программа, обремененная зависимостями? Скомпилируйте ее так, чтобы она не обращалась к другим приложениям.

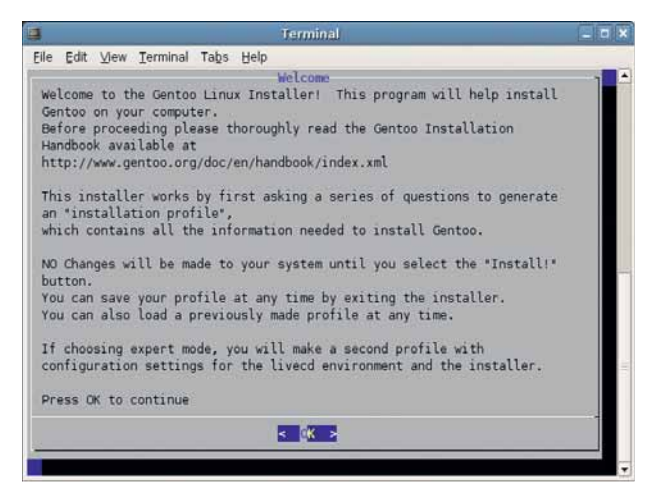


› **Рабочий стол дистрибутива Gentoo Live – Gnome цвета ванили.**

Хотите выжать из вашего компьютера максимум скорости? Оптимизируйте его под ваш конкретный CPU. Все в ваших руках!

В DVD этого месяца мы включили Live-версию Gentoo 2006.1, которая загружается в рабочий стол Gnome и позволяет произвести установку «стадии 3» (копируйте уже готовые бинарные файлы на жесткий диск). Просто загрузите ваш ПК с **LXFDVD** и нажмите Enter в меню загрузки; когда появится экран приглашения, подождите несколько секунд, и автоматически войдете в систему как пользователь 'gentoo' (кстати, вы можете создать CD-версию дистрибутива; см. **index.html** на диске). Теперь можете исследовать рабочий стол! Этот релиз включает ядро 2.6.17, Gnome 2.14.2, *OpenOffice.org 2.0.3* и *Firefox 1.5.0.5*.

Для установки на жесткий диск, мы припасли на DVD сказочно подробное руководство Gentoo Handbook, оно находится в **Distros/Gentoo/gentoohandbook.html**. Можете открыть этот документ в *Firefox* во время установки. Обратите особое внимание на раздел опций загрузки ядра, он поможет вам решить проблемы, если система стартует некорректно.



› **GUI не для героев? В Gentoo есть и текстовый инсталлятор.**

Новичкам в Linux лучше начать с супердружественной PCLinuxOS (см. следующую страницу). Но если хотите глубже копнуть вашу ОС, то все, что вам нужно для начала, здесь. Хотите получить инструкции – посетите форумы Gentoo www.gentoo.org или наши форумы на www.linuxformat.ru. »

Шаг за шагом: устанавливаем Gentoo 2006.1



1 Загрузка ›

Сохраните файл руководства по установке **gentoohandbook.html** с DVD, он вам пригодится. Загрузите компьютер с нашего диска и нажмите Enter. При возникновении проблем, обращайтесь к Руководству.



2 Запуск ›

Войдем в систему автоматически; затем дважды щелкните мышью по значку Gentoo Linux Installer (GTK+). Запустится программа установки; выберите Стандартную установку (Standard installation), или Networkless, если вы не подключены к Интернет.

3 Чтение ›

Здесь-то и понадобится Руководство: следуйте его советам и текстовым подсказкам наверху. После этого экрана вы попадете в раздел 3.c Руководства: Разбиение диска... выделите на жестком диске не менее 5 ГБ под Linux.

4 Настройка!

На стадии 13 можно выбрать готовые пакеты для установки; для рабочего стола понадобятся **xorg-x11** и Gnome. Можно добавить и другие приложения. Завершите все шаги – и вот вам отличная основа Gentoo, стройте на ней все, что угодно!

Дистрибутив Linux

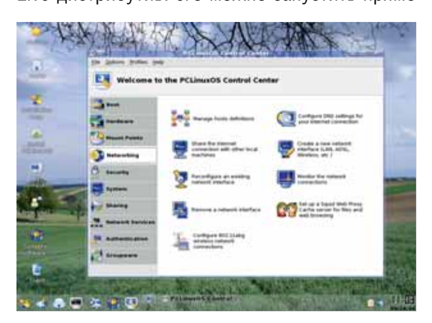
PCLinuxOS 0.93

Возьмите лучшее от Mandriva, добавьте великолепную тему рабочего стола, сдобрите постоянно растущим и готовым помочь сообществом – что у вас получится? PCLinuxOS, один из лучших новых дистрибутивов. В прошлом месяце на нашем DVD была крошечная MiniMe версия PCLinuxOS 0.93, а в этом месяце нашлось место для полного релиза 'Big Daddy' («Большой Папочка»), умело увязавшего все популярные приложения рабочего стола в одну удобную упаковку. Вы получите идеальный баланс офисного ПО, интернет-приложений, инструментов мультимедиа и даже игр. Основные программные компоненты включают ядро 2.6.16, KDE 3.5.3, *OpenOffice.org 2.0.3* и *Firefox 1.5.0.6*.

PCLinuxOS унаследовал от Mandriva превосходное распознавание оборудования, программу установки и объемистый набор инструментов настройки – хорошая основа. Разработчики добавили самый отшлифованный рабочий стол из виденных нами, а также отличную подборку программ при общем объеме менее 700 МБ – достаточно одного установочного CD. А главное, PCLinuxOS – это Live-дистрибутив: его можно запустить прямо с нашего DVD, не устанавливая на жесткий диск. Благодаря такой гибкости вы можете везде носить с собой портативный Linux, на случай, если вам вздумается подсадить пингвина в окно Windows и продемонстрировать всю мощь этой ОС, просто перезагрузив компьютер. Вдобавок PCLinuxOS очень полезен для проверки оборудования. Обидно бывает, установив полный дистрибутив, тут же обнаружить, что, к примеру, ваша видеокарта не поддерживается; но PCLinuxOS вы загрузите в считанные минуты и убедитесь, что все оборудование работает.

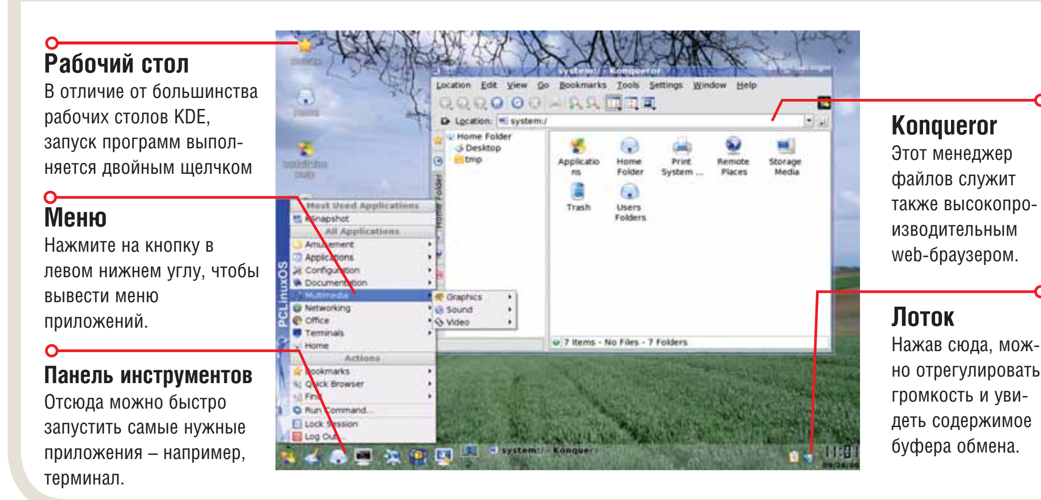
Ниже мы приводим краткое руководство по установке PCLinuxOS на жесткий диск. Процесс блаженно прост, и у вас не должно возникнуть проблем, если вы будете следовать подсказкам программы установки (понадобится не менее 256 МБ ОЗУ). Вставьте LXFDVD в компьютер и перезагрузите его (проверив, что компьютер настроен на загрузку с DVD: если надо, измените настройки BIOS). В меню загрузки два раза нажмите стрелку вниз, чтобы выбрать PCLinuxOS, и нажмите Enter. Если у вас все-таки будут проблемы, можете перезагрузить ваш ПК в безопасном режиме – Safeboot. В систему можно войти либо как гость ('guest') с паролем 'guest', либо как администратор, с паролем 'root'.

Помните также, что если вы создали CD-версию с помощью *Jigdo*, надо использовать MediaCheck, чтобы убедиться в правильности прожига. Информация по созданию CD-версии PCLinuxOS из LXFDVD содержится в **index.html** на диске. Вы всегда найдете помощь на www.pclinuxos.com и www.mypclinuxos.com



› **Настраивайте оборудование, программы, загрузку в Центре управления (Control Center).**

Знакомство с рабочим столом PCLinuxOS



Рабочий стол
В отличие от большинства рабочих столов KDE, запуск программ выполняется двойным щелчком

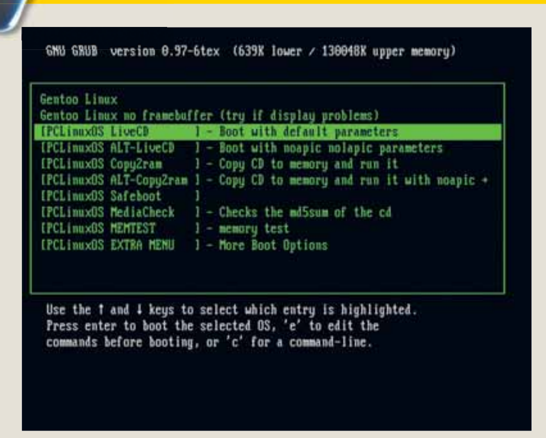
Меню
Нажмите на кнопку в левом нижнем углу, чтобы вывести меню приложений.

Панель инструментов
Отсюда можно быстро запустить самые нужные приложения – например, терминал.

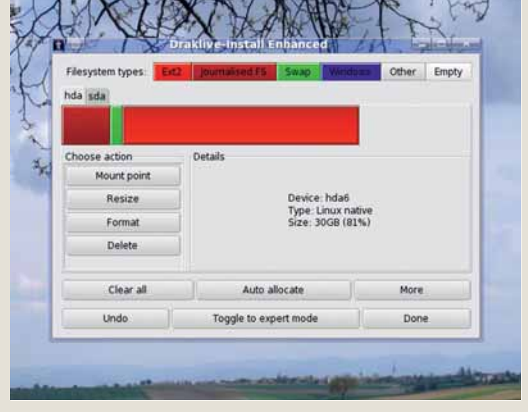
Konqueror
Этот менеджер файлов служит также высокопроизводительным web-браузером.

Лоток
Нажав сюда, можно отрегулировать громкость и увидеть содержимое буфера обмена.

Шаг за шагом: Установка PCLinuxOS



1 Вход
Загрузите ваш ПК с LXFDVD, выберите в меню PCLinuxOS; когда появится экран приглашения, войдите в систему как гость ('guest') с паролем 'guest'. Щелкните дважды по значку Install PCLinuxOS на рабочем столе и введите пароль 'root'.



2 Память
Нажмите Next, затем выберите в качестве типа установки Normal Hard Drive. Можете занять весь жесткий диск или выделить место под Linux (рекомендуется 3 ГБ) рядом с Windows. Создайте корневой раздел root (/) и раздел подкачки (не более 512 МБ).

Операционные системы

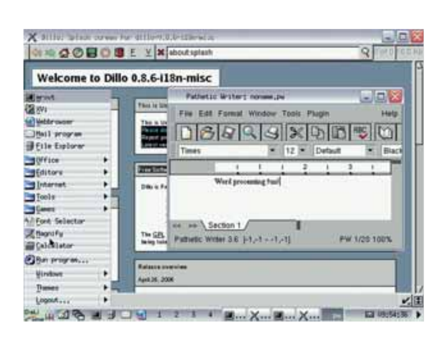
Не только Linux

Помимо крупных релизов Gentoo и PCLinuxOS, в этом месяце мы подготовили для вас подборку других дистрибутивов и ОС – пробуйте! Список открывает DeLi Linux, дистрибутив-суперлегковес, работоспособный даже на самой скромной машине. Не требуя процессора круче 486, DeLi прекрасно обойдется 32 МБ ОЗУ, а то и 16 МБ (но тут уж не ждите чудес производительности). DeLi идеален, когда нужно вдохнуть новую Linux'овую жизнь в старые машины; если у вас где-нибудь на антресолях завалялся старенький 486 или Pentium, запишите **deli-0.7.iso** на CD-R, загрузите с него компьютер и следуйте инструкциям по установке – получите маленький, но полезный рабочий стол: про запас, под сервер или для детей.

Из альтернативных систем мы представляем последние версии Haiku и ReactOS 0.3, в формате Live CD и установочного диска. Последняя операционная система, нацеленная на совместимость с программами и драйверами Windows, сделала мощный рывок и теперь справляется со многими простыми приложениями Windows, и даже со старыми версиями *Microsoft Office*. Чтобы поработать с ней, запишите **ReactOSLiveCD.iso** на CD-R и загрузите ваш ПК. Если хотите установить ее на жесткий диск (сначала сохраните все ваши данные!), используйте образ **ReactOS-install.iso**. Подробнее – в описании DVD прошлого месяца.



› **Haiku: скорость и стиль BeOS для открытого рабочего стола.**



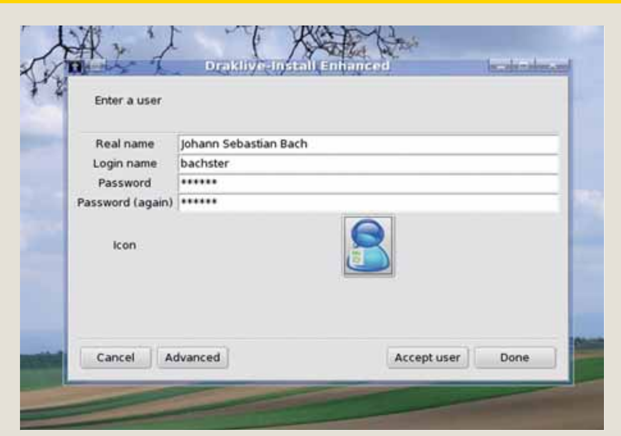
› **DeLi Linux: выведем старичков-486 из забвения.**

Haiku, клон BeOS с открытым кодом, тоже развивается молниеносно, и демонстрирует все признаки отзывчивой и простой в использовании ОС. Вы можете попробовать эту систему, скопировав файл образа жесткого диска **haiku.image** в домашнюю директорию, запустив его в эмуляторе *Qemu* (см. раздел Система) с помощью

```
qemu -hda haiku.image
```

И наконец, у нас есть один из самых ранних дистрибутивов Linux: Softlanding Linux System 0.99 (аж 1993 года рождения!). Скопируйте и извлеките **sls-0.99.tar.gz** на жесткий диск, перейдите в появившуюся директорию и запустите **./test-in-qemu**, чтобы система запустилась в эмуляторе ПК *Qemu*. Войдите в систему в качестве администратора (без пароля) и наслаждайтесь, исследуя Linux прошлых лет.

› Читайте больше об SLS на стр.24



3 Перезагрузка

Когда вы сделаете это, программа установки отформатирует разделы Linux и скопирует файлы PCLinuxOS. Примите настройки по умолчанию для программы загрузки, введите пароль администратора и создайте учетную запись обычного пользователя. Теперь перезагрузите компьютер, извлеките DVD – готово!

Документация

Журнал в формате PDF

Загляните в раздел **Журнал** на диске этого месяца: там вы найдете отличную подборку статей из прошлых выпусков журнала LXF – свыше 250 страниц в формате PDF, вы сможете их читать прямо на компьютере. Неважно, бывалый ли вы пользователь Linux, или только совершаете свои первые шаги – в любом случае вам будет куда кинуть взор. У нас три основных раздела. Первый дает вам подробную информацию по запуску собственного сервера. Нужно настроить *MySQL*? Интересуетесь доступом к файлам с помощью *BitTorrent*? Или, может, желаете испытать силы в DJing с Интернет-радиостанцией? Все это здесь есть: десять страниц информации, подсказок и руководств.

Мы также представляем статью «Дистрибутив своими руками» из LXF74, позволяющую сделать именно то, что обещано в заголовке. Если вам когда-нибудь хотелось создать собственный тематический дистрибутив с индивидуальным набором программных пакетов и установок, начните отсюда. Руководство основано на Knoppix 4, мы его смогли протащить на DVD (в раздел **Дистрибутивы**).



› **Создайте собственный Linux-тезаурус.**

Третий раздел погрузит вас в странный, полный ностальгии по прошлому мир эмуляции – вы сможете запускать на вашем ПК виртуальные версии старых консолей и компьютеров. Если вы – фанат Amiga, или вам туманят взгляд воспоминания о славных деньках SNES, здесь вы найдете способ вернуть к жизни ушедшие восьмидесятые и девяностые.

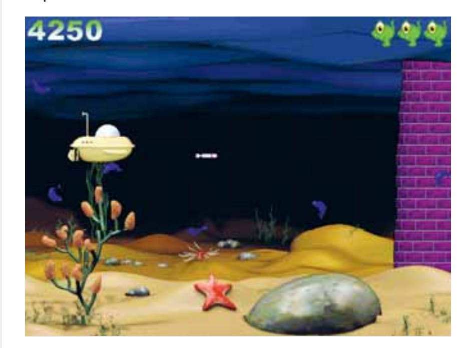
А еще – огромная подборка статей на тему «Что за штука…», объясняющих и анализирующих все технологии Linux, от Python и Jabber до планировщика ядра O(1) и KDE Solid. Если вы хоть раз вас оказывались в тупике из-за какой-то программы или протокола, больше этому не бывать: здесь найдутся все ответы.

Так что берите кофе/чай/пиво, устраивайтесь в кресле, и приятного вам чтения!

И наконец...

Несколько слов о других изюминках нашего DVD. В разделе **Рабочий стол** предлагается полный исходный код Gnome 2.16, его можно скомпилировать при помощи содержащихся на диске инструментов jhbuild и *Garnome*. В разделе **Разработка** вы найдете *Package Wizard* – приложение *Kommander*, упрощающее процесс создания бинарных пакетов. Взгляните на *Galeon* в разделе **Интернет**, если вас интересует быстрый браузер для Gnome на основе *Mozilla*.

Не забудьте и нашу обычную подборку игр: одна из них – *Freedoom* с набором свободных спрайтов, текстур и уровней для использования с движками *Doom* (например, прилагаемым *PrBoom*). Вам должен понравиться *Max Fighter*, типа классической *Asteroids* с крутыми взрывами. LXF



› ***Tower Toppler* в разделе Игры – очаровательный клон классического *Nebulus* от Hewson.**



Журнал зарегистрирован Федеральной службой по надзору за соблюдением законодательства в сфере массовых коммуникаций и охране культурного наследия
ПИ № ФС77-21973 от 14 сентября 2005 года
Выходит ежемесячно. Тираж 5000 экз.

РЕДАКЦИЯ РУССКОЯЗЫЧНОЙ ВЕРСИИ:
ГЛАВНЫЙ РЕДАКТОР
Валентин Синицын info@linuxformat.ru
Литературные редакторы
Родион Водейко, Елена Толстякова, Александр Толстой, Иван Мищенко
Переводчики
Александр Бикмеев, Светлана Кривошеина, Александр Кузьменков, Алексей Опарин, Сергей Супрунов, Александр Черных, Юлия Шабунио
Допечатная подготовка
Мария Пучкова, Родион Водейко
Креативный директор
Станислав Медведев
Технический директор
Денис Филиппов
Директор по рекламе
Денис Игнатов +7 812 965 7236 advert@linuxformat.ru
Заместитель генерального директора
Софья Виниченко
Генеральный директор
Павел Фролов
УЧРЕДИТЕЛИ
частные лица
ИЗДАТЕЛИ
Станислав Медведев, Виктор Федосеев, Павел Фролов

Отпечатано в типографии «Текст», ООО «ППК «Текст»
188680, Ленинградская область, Всеволожский район, Колтуши, д.32
Заказ _____
Пре–пресс: *d.r.i.v.e-group*

РЕДАКЦИЯ АНГЛОЯЗЫЧНОЙ ВЕРСИИ:
Редактор Ник Вейч (Nick Veitch) nick.veitch@futurenet.co.uk
Заместитель редактора Пол Хадсон (Paul Hudson) paul.hudson@futurenet.co.uk
Старший художественный редактор Мартин Парфитт (Martin Parfitt) mparfitt @futurenet.co.uk
Художественный редактор Эфрейн Эрнандес-Мендоса (Efrain Hernandez-Mendoza) efrain.hernandez-mendoza@futurenet.co.uk
Новостной редактор Майк Сондерс (Mike Saunders) mike.saunders@futurenet.co.uk
Литературный редактор
Ребекка Смолли (Rebecca Smalley) rebecca.smalley@futurenet.co.uk
Штатный автор
Грэм Моррисон (Graham Morrison) graham.morrison@futurenet.co.uk
Ассистент по выпуску
Эндрю Грегори (Andrew Gregory) andrew.gregory@futurenet.co.uk

Авторы
Ладислав Боднар (Ladislav Bodnar), Нейл Ботвик (Neil Bothwick), Д-р Крис Браун (Dr. Chris Brown), Энди Ченнел (Andy Channelle), Алекс Кокс (Alex Cox), Дэниэль Кингшот (Daniel Kingshott), Том Уилкинсон (Tom Wilkinson), Евгений Балдин, Андрей Боровский, Дмитрий Кирсанов, Тихон Тарнавский, Алексей Федорчук, Антон Черноусов, Илья Шпаньков
Художественные ассистенты: Зигги Бейкер (Ziggi Baker)
Фотографии: Джеймс Дункан Дэвидсон (James Duncan Davidson), Марк Хекстра (Mark Hoekstra), Джейсон Каплан (Jason Kaplan), Дебби Мойнихен (Debbie Moynihan)
Иллюстрации: Нейл Бартлет (Neil Bartlett), Пол Блехфорд (Paul Blachford), Отто Штейнингер (Otto Steininger), Крис Винн (Chris Winn)

КОНТАКТНАЯ ИНФОРМАЦИЯ
UK: Linux Format, 30 Monmouth Street, Bath BA1 2BW
Tel 01225 442244 Email: linuxformat@futurenet.co.uk
РОССИЯ:
Санкт-Петербург (редакция): ул. Гончарная, 23, офис 54, телефон: (812) 717-00-37
Представительство в Москве:
пр.Мира, 161, телефон +7(495) 799-18-63, +7(495)136-88-45
Email: info@linuxformat.ru, Web: www.linuxformat.ru



BATH • LONDON • MILAN • NEW YORK • PARIS • SAN DIEGO •SAN FRANCISCO



в январе

KDE 4.0
Эксклюзивный обзор

Новое возможности, свежий внешний вид и совершенно иная концепция – узнайте, как KDE заново создает рабочий стол и что это значит для вас!

О чем мечтают слоны?
Трехмерное моделирование с Blender

Mono — .NET
для сторонников Open Source



Джеф Во
Он отказался от удобного кресла в Canonical и ушел в Gnome – почему?



На диске:
Mandriva Linux 2007 и Fedora Core 6!

Они здесь, они трехмерные, они готовы побороться за ваш рабочий стол!

Содержание следующих выпусков может меняться без обязательного уведомления!

ПОДПИСКА НА LINUX FORMAT

ПОДПИСКА В ЛИНУКСЦЕНТРЕ

Сколько стоит подписка?

Подписка на журнал «Linux Format» 12 номеров (январь, февраль, март, апрель, май, июнь, июль, август, сентябрь, октябрь, ноябрь, декабрь) стоит 1800 рублей
Подписка на журнал «Linux Format» 6 номеров (июль, август, сентябрь, октябрь, ноябрь, декабрь 2006 года) стоит 900 рублей

Как оформить подписку?

Чтобы оформить подписку на журнал «Linux Format», необходимо зарегистрироваться в интернет-магазине Linuxcenter.Ru, указав ФИО и подробный почтовый адрес подписчика, заказать товар «Подписка на журнал «Linux Format» 12 номеров 2006 года», или товар «Подписка на журнал «Linux Format» второе полугодие 2006 года», получить от системы квитанцию для оплаты в любом отделении Сбербанка (для физических лиц) или счет для оплаты по безналичному расчету (для юридических лиц)

Как оплатить подписку?

– по выставленному счету (для юридических лиц)
– по квитанции в любом отделении Сбербанка

Плюсы подписки

- подписка дешевле!
- гарантированное получение нового номера журнала!

ПОДПИСКА - 2007!

ПОДПИСКА ПО КАТАЛОГАМ РФ

Каталог агентства «РОСПЕЧАТЬ» – подписной индекс 20882

Каталог «ПРЕССА РОССИИ» – подписной индекс 87974

Ф. СП-1

Министерство связи РФ

АБОНЕМЕНТ НА ЖУРНАЛ

Linux Format

ИНДЕКС ИЗДАНИЯ

КОЛИЧЕСТВО КОМПЛЕКТОВ

НА 2007 ГОД ПО МЕСЯЦАМ

1	2	3	4	5	6	7	8	9	10	11	12

КУДА

ПОЧТОВЫЙ ИНДЕКС — АДРЕС ДОСТАВКИ

КОМУ

ФАМИЛИЯ, ИНИЦИАЛЫ

ПВ — МЕСТО — ЛИТЕР

ДОСТАВОЧНАЯ КАРТОЧКА

НА ЖУРНАЛ

ИНДЕКС ИЗДАНИЯ

Linux Format

НАИМЕНОВАНИЕ ИЗДАНИЯ

СТОИМОСТЬ: ПОДПИСКИ ___ РУБ. ___ КОП.; ПЕРЕАДРЕСАЦИИ ___ РУБ. ___ КОП.

КОЛИЧЕСТВО КОМПЛЕКТОВ

НА 2007 ГОД ПО МЕСЯЦАМ

1	2	3	4	5	6	7	8	9	10	11	12

КУДА

ПОЧТОВЫЙ ИНДЕКС — АДРЕС ДОСТАВКИ

КОМУ

ФАМИЛИЯ, ИНИЦИАЛЫ



ПОДПИСКА НА LINUX FORMAT

ПОДПИСКА ПО КАТАЛОГАМ СНГ И БЛИЖНЕГО ЗАРУБЕЖЬЯ

Каталог «Российская Пресса» – совместный проект Государственного предприятия «Казпочта», Агентства «Книга-Сервис» и АРЗИ.

Блок изданий АРЗИ в национальных Каталогах Украины и Беларуси. В Азербайджане, Армении, Грузии, Киргизии, Узбекистане и Молдове - по изданиям, включенным в Объединенный каталог, распространяемые через АРЗИ.

Азербайджан

- по Объединенному каталогу российских изданий через Предприятие по распространению печати «Гасид»(370102, г. Баку, ул. Джавадхана, 21);

Армения

- по списку номенклатуры «АРЗИ» через ГЗАО «Армпечать» (375005, г.Ереван, пл.Сасунци Давида, д.2) и ЗАО «Контакт-Мамул» (375002, Г.Ереван, ул.Сарьяна, 22);

Белоруссия

- по Каталогу изданий стран СНГ через РГО «Белпочта» (220050, г.Минск, пр-т Ф.Скорины, 10);

Грузия

- по списку номенклатуры «АРЗИ»» через АО «Сакпресса» (380019, г.Тбилиси, ул.Хошараульская, 29) и АО «Мацне» (380060, г.Тбилиси, пр-т Гамсахурдия, 42);

Казахстан

- по Каталогу «Российская Пресса» через ОАО «Казпочта» и ЗАО «Евразия пресс»;

Молдавия

- по каталогу через ГП «Пошта Молдавей» (МД-2012, г.Кишинев, бул.Штефан чел Маре, 134);
- по списку через ГУП «Почта Приднестровья» (MD-3300, г.Тирасполь, ул.Ленина, 17);
- по прайс-листу через ООО Агентство «Editil Periodice» (2012, г.Кишинев, бул. Штефан чел Маре, 134).

Узбекистан

- по Каталогу «Davriy nashrlar» российские издания через Агентство по распространению печати «Davriy nashrlar» (7000029, Ташкент, пл.Мустакиллик, 5/3, офис 33);

Украина

- Киевский главпочтамт.
- Подписное агентство «KSS» Телефон/факс (044)270-62-20, 270-62-22

ПОДПИСКА НА LINUX FORMAT

Агентство “Centerpress”

Сколько стоит подписка?
Подписка на журнал "Linux Format" 12 номеров (январь, февраль, март, апрель, май, июнь, июль, август, сентябрь, октябрь, ноябрь, декабрь 2007 года) стоит 1800 рублей.

Как оформить подписку?
Чтобы оформить подписку на журнал "Linux Format", необходимо зарегистрироваться в интернет-агентстве Centerpress.ru, указав ФИО и подробный почтовый адрес подписчика, заказать товар "Подписка на журнал "Linux Format" на 2007 год 12 номеров (01-12 / 2007)", получить от системы квитанцию для оплаты в любом отделении Сбербанка (для физических лиц) или счет для оплаты по безналичному расчету (для юридических лиц)

Агентство “Centerpress”: www.centerpress.ru

Все Плюсы подписки!
- Подписка дешевле!
- Гарантированное получение журнала!

По каталогам РФ

Каталог агентства “РОСПЕЧАТЬ” - подписной индекс

20882

Каталог “ПРЕССА РОССИИ” - подписной индекс

87974



АЛЬТЕРНАТИВНЫЕ АГЕНСТВА РФ

Агентство «Интер-Почта» (095) 500-00-60, курьерская доставка по Москве.

Агентство «Вся Пресса» (095) 787-34-47

Агентство «УралПресс»

- Екатеринбург, Березовский, В. Пышма, Первоуральск
  тел. (343) 375-80-71, 375-84-93, 375-84-39,
  факс 375-62-74,
  info@ural-press.ru
- Нижний Тагил
  тел. (3435) 411448, 417709,
  ntagil@ural-press.ru
- Челябинск
  тел. (351) 262-90-03, 262-90-05,
  pochta@chel.surnet.ru
- Пермь
  тел. (3422) 60-24-40, 60-22-95, 60-35-42,
  parma-press@permonline.ru